http：//www.readfree.net

网上读书园地酷马车技术帖汇集

主要技术帖子来源于“酷马车”，即 coolman、马健、cheming 在论坛发的非加密帖（谁说有技术含量的帖子都加密了？？），主要为马健大侠的技术帖，还有部分是我以前保存下来的，来源于其他朋友的帖子，很抱歉不记得 id 了，也懒得去查，感谢技术牛人们的无私奉献。

本技术汇集帖适合新手，以及比较繁忙，偶尔才来一次论坛的朋友，老手也可以下一个放家里备用，呵呵。整理时只考虑了内容，没有精力也没有能力搞美工，相信多数需要的朋友也不会在意美工的：）

scdymy 整理　2008 年 1 月 3 日

目　录

coolman早期写的置顶帖 3
DjVu转PDG的方法与步骤 5
用DjVuToy v0.11 玩转DjVu中的隐藏文字 6
在简体中文Office 2003 下OCR繁体中文、日文、韩文 7
PDG转PDF注定会文件膨胀、质量下降吗？ 12
PDG转图像、PDF的若干方法 16
说“DPI” 21
PDG文件的DPI 27
说“层” 28
图像转PDF的问题、方法及题外话 32
文本PDG文件名构成 54
文本PDG转PDF 55
文本文件合并批处理 56
给清晰版PDG无损减肥 58
关于去除JPG水印后质量下降的问题 59
吵醒文件加密方式说明[车大师开讲] 60
关于文本超星的字体 65
对bookinfo.dat的说明 66
对文本PDG格式的争论可以暂停了 68
乱谈zip、rar文件格式 69
用Pdg2.DLL解码PDG的境界 74
用PDG的瓶子，装PNG的酒 76
InfoRule.dat 的解密算法 - 答 flyfox 77
PDG下载软件不完全比较 78
破解超星打印量限制 80
手工计算有试读（有封面）ss号的方法： 81
用wget, awk批量获得超星图书书目 83
去除JPG水印的简易方法(10 月 30 日第二次重大修正) 84
用wget也可以实现多线程下载 85
pcookie.htm 86
CX入口的搜索方法 88
主站有没有某书用主站客服教大家的方法 88
wget--强大的下载工具 89
readfree新手推荐赚钱路线 (其实很有技术含量,非水帖)[dingdangch] 90
scdymy搜集的新手入门帖子 93

http://www.readfree.net/bbs/read.php?tid=152429

coolman早期写的置顶帖

作者：coolman

二、技术问题 【资源相关】

1. 什么是超星文件格式？ 如何判别？

超星文件一般以 pdg 为扩展名，分文本和图像 2 种。一般所指的格式是针对图像类的。 _

可以用任何 16 进制文件编辑器打开，如 ultraedit, 第 16 个字节处的数字就是该文件的类型。例如 02H 表明该处数字为 02. Jpeg 格式除外。

2. 超星文件格式有哪些？

现有格式 00－05h, 10-1CH, 64-68h, aah,abh,ach, FFh(其中 FFh 格式为已经破坏的格式，无法阅读）以及 Jpeg 格式

看下图

[attachment_01=30268]

3. 超星文件会过期吗？ 如何避免？

10h, 64-68h 的文件均为加密格式，有阅读限制。避免的办法：1. 避免得到这样的文件。2. 买卡并在过期前备份注册信息，阅读时导入注册信息，并断开网络连接。3. 过期后续卡。4. 注册 mini pdg reader, 并申请 reader key.

4. 超星文件可以 转换为其它图像格式吗？

可以。1. 虚拟打印是最简单的办法 2. pdg2bmp 程序 3. Pdg2Pic 程序

5. 超星文件可以打印吗？

可以，但是不同版本的限制不同。 寻找破解或者参考回答 4。

6. 为什么有些 05h 的文件 SSREADER 不能阅读？

排除文件下载不完整的因素后，最可能的原因是该文件为 DJVU 格式的，如下图，只需要将 05 改为 00 即可。 └

[attachment_01=30273]

7. ssreader 3.9 下载的本地文件会被破坏吗？ 如何避免？

用 ssreader 3.9 阅读本地文件时会判断该文件是否过期，如果过期，该文件所在目录的所有本地文件都会被破坏，这样的文件都有一个日期标志，可以清除此标志，避免被破坏。看下图

[attachment_01=30278]

8. be 下载的某些 04h,05h 的 pdg 文件不能阅读，怎么办？

这样的文件一般偏移为 66h 的字节为 01，修改为 00 即可。

（最新版 mini pdg reader 已经可以直接阅读）

参看

http://www.readfree.net/bbs/read.php?tid=152046

9. 直接能在 SSREADER 中无需注册就能阅读的图书格式都有哪些？

最新的 SSREADER 版本可以直接阅读的格式为 00,02,03,04,05 和 JPEG 格式

10. 如何在 SSREADER 中阅读 1xh 和 axh 格式书？

某些（不是全部）1xh 和 axh 格式的书可以通过建立本地虚拟 http web 服务

器来阅读。最简单的虚拟 http web 服务器是时空 web 软件。或者用认证区的 pdgserver 阅读，pdgserver 应该支持目前所有的 1xh 和 axh 格式。

11. pdg 如何转换为 pdf?

有 4 种办法：

a. 直接虚拟打印，可以用各种 pdf 的虚拟打印程序，但是要把打印机改名。此法方便快速，但是会降低原始图片的分辨率。解决办法是寻求破解。

b. 利用 pdg2bmp&jpg&tif&pdf&txt 直接转换，此法不会降低分辨率，但是速度慢，而且不是所有格式都支持。

c. 利用 boox viewer 或者 pdg2bmp&jpg&tif&pdf&txt 先转换为 bmp， 然后将 bmp 转换为 pdf. 此法的优缺点同 b.

d. 利用 strnghrs 的 Pdg2Pic 程序和 FreePic2Pdf 程序直接完成，此方法简单快捷，不损失图像质量，也不增加文件大小，推荐！

12. 如何避免 ssreader 3.9 的超过本月打印限制?

a. 备份安装目录下的 ssreader.ul，或将安装目录下的 ssreader.ul 设为只读。若提示打印页数已满，用原来备份的 ssreader.ul 覆盖掉原来的即可。（据会员发贴整理，版主没有验证。）

b. 寻找补丁或者破解， 如 ssloader

13. 1xh 的 pdg 格式可以转换为 02h 格式吗？

可以。可以转换的软件有 1xhkiller(已经删除，不再提供)，pdg11 和 boox viewer（部分不支持），注册版 mini pdg reader.

14. axh 的 pdg 格式可以转换为 05h 格式吗？

可以。可以转换的软件有 axhkiller(仅仅支持部分 axh 格式，已经删除，不再提供)，注册版 mini pdg reader. 另外 cheming 的 Pizza v1.0 Lite(解码 pdg 格式的软件)可以将 1xH,axH, 04H 转换为原始的 00H 格式，推荐！

15. 6xh 的 pdg 格式可以转换为 02 或者 05h 格式吗？

可以。可以转换的软件有注册版 mini pdg reader. 另外 cheming 的 Pizza Pro v1.3(解码 pdg 格式的软件)可以将 1xH,axH, 04H, 6xH 等转换为原始的 00H 格式，推荐！

16. 如何避免得到 FFh 格式？

FFh 格式的原始格式为 axh 格式，一般可以在线阅读，用 ssreader 下载后变成 FFh 格式，文件头被破坏，不能阅读，也不能修复。为避免得到 FFh，可以在 ssreader 在线阅读用网络嗅探软件截留；可以采用第三方软件如 BE 下载；可以采用破解版本的 ssreader 下载。本版不负责提供相关软件，也不回答如何截留等问题。

17. pdg 可以在线阅读但是不能下载，怎么办？

答案同 16。

18. 6xh 加密格式的 pdg 如何直接在不同机器阅读？

方法 1：购买 mini pdg reader,然后申请 reader key

方法 2：购买 ssloader

方法 3：购买 mini pdg reader professional

19. 不同版本的超星阅读器能安装和运行在一台电脑上吗？

可以。但安装时一定要注意不要覆盖文件，因为超星阅读器使用的某些文件可能均放在系统目录下。有些还必须注册才能使用。要使得多个版本都能正常切换运行，最好的办法是到论坛寻找多版本启动器。

http://www.readfree.net/bbs/read.php?tid=4477073&keyword=

DjVu转PDG的方法与步骤

作者：马健

声明：

1、谨以此文献给喜欢折腾的各位热血人士，不喜欢折腾的就不必看了。

2、本文欢迎转载，不过转载的时候请注明原作者为 strnghrs。

3、DjVu 转换成 PDG 后，打开可能会有点慢：既然在空间上赚取了利润，在时间上付出一点成本也是应该的。

一、准备散页 DjVu

怎么获得 DjVu 文件就不必问我了，问了也不会有结果。

如果获得的是打包后的多页 DjVu，可以用 DjVuToy 的“文件拆分”功能拆开。

二、文件更名

散页 DjVu 需要更名为 PDG，并且符合 PDG 文件名规范：主文件名为 6 位字母、数字，控制名位 pdg，均为小写。

主文件名由前缀加数字组成，前缀含义为：

cov：封面

bok：书名

leg：版权

fow：前言

!：目录

att：附录

bac：封底

ins：插页

正文页无前缀，直接用 6 位数字编码。

更名工具很多，我习惯用 RenameIt。如果有人做个专用工具，估计能赚点论坛币出来。

三、转成真正 PDG 文件

PDG 文件本身是支持 DjVu 压缩的，只是需要在前面加上 PDG 文件头，所以转换完成后，文件总长度会比原 DjVu 文件总长度大一点。

转换方法：用 DjVuToy 的“PDG 压缩”功能，选择上一步中名为 PDG，实为 DjVu 的文件所在文件夹，注意不要选“转换为快速版”，这样可以保证最大限度保持清晰度。

对于黑白单层 DjVu（只有 Sjbz 段，无 FG44、BG44、FGbz 等），DjVuToy 会在 PDG 文件头后直接嵌入原 DjVu 文件，实现无损转换。对于灰度、彩色 DjVu（含 FG44、BG44、FGbz 等段），由于 PDG 浏览器对这类文件的解释与众不同（上下颠倒、颜色互换），所以只能先解码，再重新压缩成单层 DjVu（只含 BG44），因此文件质量或长度可能会有一点损失。

http://www.readfree.net/bbs/read.php?tid=4537397&keyword=
作者：马健

用**DjVuToy v0.11** 玩转**DjVu**中的隐藏文字

Q：OCR 功能有什么用？在什么情况下可以使用？

A：OCR 功能在 DjVu 文件中生成隐藏文本，这些文本平时不可见，但可用 WinDjVu 的“Edit->Find”功能检索，也可以用“File->Export Text”功能导出。隐藏文本不仅有文本信息，而且有位置信息，因此用鼠标按住左键在 DjVu 页面上拖动，可以选中隐藏文字，并复制到剪贴板。

DjVuToy 的 OCR 功能对 DjVu 中的原始图像不会造成任何影响，因此可以对其它软件生成的 DjVu 文件进行 OCR，以实现强强联合：目前 DjVu 制作软件以国外的为佳，但是国外 DjVu 制作软件在 OCR 中文时总觉得不如本土软件。DjVuToy 的 OCR 引擎是微软从清华购买的，中文 OCR 效果不错。

当然再好的 OCR 软件都不可能完全准确，因此 DjVuToy 提供了独创性的“导出 XML 文本”、“导入 XML 文本”功能，可以将隐藏文本及其位置信息以 XML 格式导出，进行人工校对，然后再导入 DjVu 文件。 另外这两个功能也可以用于文本的繁简转换：将繁体导出，用 TextForever 或其它转码软件转成简体，然后再导入。

当然如果您有更好的 OCR 引擎，也可以自己写一个软件，OCR 后输出符合 DjVuToy 格式要求的 XML 文件，然后用 DjVuToy 导入。

DjVuToy 的 OCR 功能需要微软 Office 2003 以上版本的 Microsoft Office Document Imaging 的支持，对于 Office 2003、2007，这个功能可能缺省安装都没有装全（Office 2007 的缺省安装干脆就没装），需要补充安装。

在简体中文环境下 OCR 繁体中文，及在繁体中文环境下 OCR 简体中文的方法，可以 google 我写的《用 Pdg2Pic、TextForever 实现批量 OCR》一文。

更多说明见 DjVuToy 使用说明。

新版售价小涨至 75 币，当年 30 币买入的可以偷着乐了。

为了增加说服力，增加一个例子：从中美百万下载的繁体、竖排明河版《倚天屠龙记》。“原版”指的是百万版，明显可以看出是把繁体当作简体 OCR；“简体”是用 DjVuToy 进行繁体 OCR->导出->繁体转简体->导入，可以看出错别字少多了；“校对”则是将“简体”的文字导出，校对，然后再导入，基本上没有错别字。

http://www.readfree.net/bbs/read.php?tid=4550673&keyword=

在简体中文Office 2003 下OCR繁体中文、日文、韩文

作者：马健
邮箱：stronghorse@tom.com
主页：http://stronghorse.yeah.net
发布：2007.12.08

目录

一、引子
二、系统配置
  1、原理
  2、实战
    繁体中文配置
    日文配置
    韩文配置
    简体中文配置
三、其他讨论

一、引子

在简体中文 Office 2003 下用 Micorsoft Office Document Imaging (MODI)做 OCR 的步骤为：

先确保 MODI 已经正常安装。Office 2003 的缺省安装是第一次使用 MODI 时安装，Office 2007 的缺省安装是不装，都需要改过来。

在资源管理器里选中某个多页 TIFF 文件，从右键菜单选择用 Micorsoft Office Document Imaging 打开。

打开后，先选择“工具->选项”，对 OCR 选项进行设置。常规设置是去掉“自动拉伸”、“自动旋转”选项，再选择合适的语言。

选择“工具->将文本发送到 Word”，在弹出的对话框中选择“所有页面”，“在输出时保持图片版式不变”，然后选择默认文件夹，点“确定”，即可开始 OCR。

OCR 结束后，文本自动发送到 Word。缺省格式是 HTML，当然也可以另存为 txt、doc。

与其他商业 OCR 软件相比，MODI 具有下列特点：

支持多页 TIFF。某些 OCR 只支持单页 TIFF，OCR 以后还需要对结果进行合并。当然 MODI 支持的 TIFF 页数也不是无限的，我个人的经验是不要超过 300 页。单页 TIFF 文件可以用免费的 TiffToy 合并成多页 TIFF，然后再用 MODI 进行 OCR。TiffToy 合并时可以选择每合并多少个文件生成一个新文件。

中文标点、文本段落保持得比较好，后期校对省了很多事。

支持的语言比较多，Office 支持的语言基本都支持。但是这一点对大多数用

户来说无法体会，因为正常情况下，MODI 只支持英文和当前 Office 语言（如简体中文）的 OCR，要想支持更多的语言，需要进行一些设置，这就是本文所要讨论的内容。当然我并非语言天才，对于亚洲主要语言（中、日、韩）还算有所了解，其他语言一概无知，所以本文的讨论也仅限于这三国语言。

提供开放的编程接口。对于软件开发人员来说，到微软网站下载一份 MODI 编程手册，即可开发出基于 MODI 的、具有多国语言 OCR 功能的软件。

在正式开始讨论系统设置前，先透露一点技术背景：

MODI 所使用的中、日、韩 OCR 引擎，均为清华文通的 OCR 引擎。

由于简体中文平台的 GBK 字符集完全覆盖繁体中文、日文，因此繁体中文、日文的 OCR 结果在简体中文 Office 环境下均为 GBK 编码，可以在支持 GBK 编码的中文平台下正常显示、编辑。当然如果觉得繁体中文看起来比较麻烦，也可以用 Word 的繁简转换功能，或 TextForever 的编码转换功能，将 GBK 繁体转换成 GB 编码的简体。但是对于韩文来说就没有这么美好了，因为目前 GBK 还不兼容韩文，所以韩文的 OCR 结果如果想在简体 Office 下编辑，大概只能存为 HTML 或 doc 文件，然后用 Word 编辑。

MODI 编程手册可以到这里下载：

http://www.microsoft.com/downloads/details.aspx?FamilyId=8F93E445-B1CF-4477-A373-E17417D616BC&displaylang=en

二、系统配置

1、原理

要想让简体中文 Office 2003 能够 OCR 繁体、日文、韩文，需要做的工作包括两个方面：

安装相关语言的 OCR 模块。MODI 本身可以看作一个外壳，真正的 OCR 功能需要靠不同语言的模块实现。每个语言模块包括相关 DLL 文件和数据文件，需要复制到 MODI 的安装文件夹下。

告诉 MODI，目前有哪些语言的 OCR 模块可以使用。这个需要更改注册表，更改后在 MODI 的 OCR 选项里即可选择对应的语言。

2、实战

繁体中文配置

找一台安装了繁体中文 Office 2003 的机器，进入 MODI 的安装文件夹，缺省为：

C:\Program Files\Common Files\Microsoft Shared\MODI\11.0

将下面的文件复制到安装了简体中文 Office 2003 的相同文件夹下：

TCCODE.UNI

TCPRINT.DAT

TCPRINT2.DAT

TCSERHT.DAT

TCTREE.DAT

TW_BU.DAT

TW_UB.DAT

TWBIG532.DLL

复制完成后，用记事本创建一个 reg 文件，把下面内容粘贴后存盘：

Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Installer\Components\61BA386016BD0C340BBEAC273D84FD5F]

"1028"=hex(7):28,00,26,00,48,00,42,00,56,00,6e,00,2d,00,7d,00,66,00,28,00,5a,\
00,58,00,66,00,65,00,41,00,52,00,36,00,2e,00,6a,00,69,00,4f,00,43,00,52,00,\
5f,00,31,00,30,00,32,00,38,00,3e,00,7d,00,60,00,45,00,4d,00,61,00,65,00,2c,\
00,37,00,71,00,39,00,2a,00,44,00,58,00,64,00,55,00,40,00,45,00,50,00,69,00,\
3d,00,00,00,00,00

双击此 reg 文件导入注册表后，在 MODI 的 OCR 选项卡里，“OCR 语言”即可看到“中文（繁体）”。注意导入注册表时必须先关闭所有 MODI 窗口，导入后再打开。

在简体中文环境下，按照上述步骤设置后，用 MODI 识别出来的繁体中文是 GBK 编码的繁体字，可以用 Word 的繁简转换，或 TextForever 的编码转换功能 （支持批量）转换成 GB 编码的简体字。

日文配置

需要从日文 MODI 复制到简体 MODI 文件夹下的文件为：

JPCODE.UNI

JPPRINT.DAT

JPPRINT2.DAT

JPSERHT.DAT

JPTREE.DAT

TW_SU.DAT

TW_US.DAT

TWRECJ.DLL

TWSJIS32.DLL

需要导入的 reg 内容为：

Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Installer\Components\61BA386016BD0C340BBEAC273D84FD5F]

"1041"=hex(7):30,00,5d,00,67,00,41,00,56,00,6e,00,2d,00,7d,00,66,00,28,00,5a,\
00,58,00,66,00,65,00,41,00,52,00,36,00,2e,00,6a,00,69,00,4f,00,43,00,52,00,\
5f,00,31,00,30,00,34,00,31,00,3e,00,2e,00,61,00,45,00,4d,00,61,00,65,00,2c,\
00,37,00,71,00,39,00,2a,00,44,00,58,00,64,00,55,00,40,00,45,00,50,00,69,00,\
3d,00,00,00,00,00

配置成功后，在 MODI 的 OCR 选项卡里，“OCR 语言”即可看到“日语”。

在简体中文环境下，按照上述步骤设置后，用 MODI 识别出来的日文是 GBK 编码，可以在支持 GBK 字符集的简体中文环境下正常显示、编辑。

韩文配置

需要从韩文 MODI 复制到简体 MODI 文件夹下的文件为：

DATASIM.DAT
HANGULLB.DAT
KRCODE.UNI
KRDIST.DAT
KRPRINT.DAT
KRSERHT.DAT
KRTREE.DAT
TW_KU.DAT
TW_UK.DAT
TWCUTCKR.DLL
TWCUTLKR.DLL
TWKSC32.DLL
TWLAYKR.DLL
TWRECK.DLL

需要导入的 reg 内容为：

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Installer\Components\61BA386016BD0C340BBEAC273D84FD5F]
"1042"=hex(7):31,00,5d,00,67,00,41,00,56,00,6e,00,2d,00,7d,00,66,00,28,00,5a,\
00,58,00,66,00,65,00,41,00,52,00,36,00,2e,00,6a,00,69,00,4f,00,43,00,52,00,\
5f,00,31,00,30,00,34,00,32,00,3e,00,30,00,61,00,45,00,4d,00,61,00,65,00,2c,\
00,37,00,71,00,39,00,2a,00,44,00,58,00,64,00,55,00,40,00,45,00,50,00,69,00,\
3d,00,00,00,00,00
```

配置成功后，在 MODI 的 OCR 选项卡里，“OCR 语言”即可看到“朝鲜语”。

在简体中文环境下，按照上述步骤设置后，用 MODI 识别出来的韩文是韩文编码（charset:129），可以存为 HTML、doc，并能在 Word 里正常显示、编辑。如果存为 TXT，则不能在简体中文环境下显示、编辑。

简体中文配置

如果需要在繁体中文环境下 OCR 简体中文，最正宗的方法是下载、安装一个简体 MODI：

http://www.microsoft.com/downloads/details.aspx?familyid=dd172063-9517-41d8-82af-29c38f7437b6&displaylang=zh-tw

当然如果想省事，也可以复制下列文件：

SCCODE.UNI
SCPRINT.DAT
SCPRINT2.DAT
SCSERHT.DAT
SCTREE.DAT
TW_GU.DAT
TW_UG.DAT
TWGB32.DLL

需要导入的 reg 内容为：

Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Installer\Components\61BA386016BD0C340BBEAC273D84FD5F]
"2052"=hex(7):4d,00,6a,00,33,00,47,00,51,00,66,00,5e,00,62,00,54,00,3f,00,42,\
00,3f,00,56,00,50,00,24,00,5e,00,62,00,53,00,6c,00,6c,00,3e,00,25,00,6d,00,\
45,00,4d,00,61,00,65,00,2c,00,37,00,71,00,39,00,2a,00,44,00,58,00,64,00,55,\
00,40,00,45,00,50,00,69,00,3d,00,00,00,00,00

三、其他讨论

详见《用 Pdg2Pic、TextForever 实现批量 OCR]》。

http://www.readfree.net/bbs/read.php?tid=202214&keyword=

PDG转PDF注定会文件膨胀、质量下降吗？

作者：马健
邮箱：stronghorse@tom.com
主页：http://stronghorse.yeah.net
发布：2006.07.16
更新：2006.07.20

事先声明：

PDG 文件是超星公司电子图书的专有格式，需要用超星公司的专用浏览器才能阅读。本文讨论 PDG 转 PDF 的方法，仅出于技术研究目的，并无意对超星公司的版权进行任何形式的侵犯，也不希望任何人用本文讨论的工具或方法从事侵权活动。如果需要浏览 PDG 电子书，请通过购买点卡等方式，以合法的途径获得。本文 认为用户通过合法的手段获得 PDG 文件，只是由于希望能够在比超星浏览器更好、更方便的浏览器上阅读，并且不对转换出来的文件进行扩散的情况下，才需要将 PDG 文件转换成 PDF 文件。

本文所说 PDG，是指最常见的纯图像格式 PDG 文件，不包括罕见的文本、PDF、HTML 等格式的 PDG 文件。

对于标题所提问题，我的回答是：大多数情况下转换后文件长度应该相当，略有增加或减少，质量应该保持不变。如果文件长度、画面质量差很多，多半是用错了方法或软件。本文将说明理由。

注意在上面的回答中我使用了“大多数”等表示概率的词汇，因此在讨论答案之前，需要先对 PDG 文件格式进行分类，并估计每种 PDG 文件出现的可能性。

目前比较流行的分类方法是按照 PDG 文件第 16 字节分类，通常 PDG 文件格式检查软件都是按照这个字节的 16 进制报告文件类型，如 00H、02H、03H、04H、05H、10H、11H、AAH、ACH、64H、66H 等等。由于这种分类法分出来的类型较多（理论上有 256 种），所以通常也按字节高 4 位进行归类，简称 0xH、1xH、AxH、6xH 等。

这种分类方法可以表示出 PDG 文件的加密特征：

00H 是最早，也是最原始的 PDG 格式，其格式为：PDG 文件头＋原始图像数据流。原始图像数据流包括 CCITT G4（黑白图像）、JPG（彩色/灰度）、DjVu（黑白/彩色/灰度）。在超星服务器上，这种格式的文件已经非常少见，但是由于这种格式阅读的时候不需要解密，因此阅读 时的速度感觉比其它格式的要快，所以也有人用第三方软件自己将其它格式转换成 00H 格式。

0xH 是对 00H 的弱加密格式，通常 02H、03H 用来加密 CCITT G4 图像，04H

加密 JPG，05 加密 DjVu。顺便说一句，可能是为了尽量减小文件长度，超星在压制 DjVu 时，用的都是有损压缩，可能会对汉字笔划造成损伤，这也是为什么经常听到有人说 05H 不如 02H 清晰的原因之一。

1xH 是比 0xH 更强的加密，加密方法不再与原始图像格式对应，如 11H 可以加密 CCITT G4，也可以加密 JPG、DjVu。

AxH 的加密强度比 1xH 更强，加密方法也不与原始图像格式对应。正版超星浏览器如果下载到 AxH 格式的 PDG，会将其完全破坏后变成 FFH 格式的 PDG。

6xH 是正版超星浏览器从服务器下载到 PDG 文件后在本地加密生成的文件。由于 6xH 加密使用了本地硬盘“指纹”，因此只能在下载的机器上看，换一台就不能看。

但是加密方法毕竟是超星自己的事，用这种分类方法往 PDF 格式上套未免有点难。所以我更愿意用另一种分类标准：超星浏览器自带的 Pdg2 控件的 GetImageType 方法的返回值。这个方法通常返回 1、2、3，在 Pdg2Pic 中分别用 T1、T2、T3 表示，即 Type1、Type2、Typ3 的意思，分别对应三种图像：

T1：黑白图像，原始图像格式为 CCITT G4 或 DjVu。

T2：灰度/彩色图像，原始图像格式为 JPG 或 DjVu。可能超星觉得扫描时区别灰度、彩色太麻烦了，所以灰度图像一律按彩色存储，这 也是国内扫描外包商的通常做法，但是对技术较真的客户一般会要求外包商举行区别。

T3：多层图像，底层黑白文字层通常用 CCITT G4，上层插图用 JPG。我猜测这种类型应该是在 DjVu 基础上发展出来的，符合“按需存储”的原则：对于重要的文字层使用无损压缩，对于相对不重要的插图则用有损压缩存储。

从出现的概率来说，这三种格式按从高到低排列依次是 T1、T3、T2：

最常见的格式还是 T1，毕竟大多数书籍都是白纸黑字。

T3 出现的概率比 T1 小，一般用于图文混排的插图页，或某些彩印书籍。

T2 出现的概率最小，毕竟除了封面和某些特殊书籍外，整页都是图的情况在一本书里也不会有几页，而封面、封底还有很大一部分直接用 JPG 文件存储。

在转换成 PDF 时，正常情况下这三种格式与 PDF 中压缩算法的对应关系为：

T1：CCITT G4 或 JBig2。

T2：JPEG 或 JPEG 2000。

T3：这个比较复杂，取决于转换软件：可以转换成多层 PDF，也可以将 PDG 中的所有层合并成一个图像再放入 PDF。

如果 T1 原始图像是 CCITT G4，转换成 PDF 的 CCITT G4 后文件尺寸会略有膨胀，因为 PDF 文件本身要增加一些必要的格式信息；如果转换成 PDF 的 JBig2，通常文件尺寸不会增加，只会减少，毕竟 JBig2 的压缩算法要比 CCITT G4 更先进。

如果 T1 原始图像是 DjVu，转换成 PDF 的 CCITT G4 后文件无疑将会膨胀；转换成 JBig2 则取决于是有损 JBig2 还是无损 JBig2。理论上说，DjVu 对黑白图像的压缩能力与 JBig2 相当，但由于超星用的全是有损 DjVu，因此转换成 PDF 时只有选有损 JBig2 才能保持二者尺寸 大致相当，选无损 JBig2 则会造成文件膨胀。由于有损压缩会对汉字笔划造成损伤，因此我宁愿文件长度膨胀，也不愿

选择有损。

如果 T2 原始格式是 JPG，转换结果取决于转换软件：如果转换软件能够直接从 PDG 文件中提取原始 JPG 数据流嵌入 PDF，则 PDF 文件只会略有膨胀 ，质量不变；如果转换软件非要把 PDG 先解码成 BMP 再压缩成 JPG 或 JPEG 2000 放到 PDF 里，文件长度可能增加也可能减小，取决于所选的压缩比，但是在缺省的压缩比下，质量下降是注定了的。

如果 T2 原始格式是 DjVu，用于目前的 PDF 规范不支持 DjVu（不排除将来会支持），因此只能将 DjVu 先解码成 BMP 再压缩成 JPG 或 JPEG 2000 放到 PDF 里，文件长度可能增加也可能减小，质量多半会下降。

对于 T3，如果转换软件能够直接从 PDG 文件中提取原始 JPG 数据流嵌入 PDF 文件，并且用无损 JBig2 压缩原 CCITT G4 图像，则 PDF 文件尺寸会减小，同时质量不变；如果转换软件非要把多层合并成一层，再压缩成JPG或JPEG 2000 放到 PDF 里，通常文件长度会增加，质量会下降：JPG 或 JPEG 2000 都不适合压缩文字图像。

综上所述，从原理上说，对于最常见的黑白 PDG，转换成 PDF 后应该长度略有减少，质量保持不变；对于带插图的多层 PDG，转换成 PDF 后应该长度略有减少，质量保持不变；对于纯图像页面的 PDG，转换后长度可能略有增加而质量不变，也可能长度、质量都有较大变化，但是这种页面毕竟不多。

现在各位明白我在本文开始部分给出的回答的含义了吧？

在明白的同时，我相信也会有人合理地引伸出另外一个问题：为什么现在大家看到从 PDG 转出来的 PDF，会和原始 PDG 差那么多？

我认为这方面最大的罪魁祸首就是广为流传的“打印大法”：将 PDG 文件直接从超星浏览器打印到虚拟 PDF 打印机。这种方法的制约因素我已经在《PDG 转图像、PDF 的若干方法》一文中加以说明，对转换出来的 PDF 举行分析所需的工具和方法，也在《图像转 PDF 的问题、方法及题外话》一文中详细说明，喜欢较真的人不妨自己验证，这里我只说我的结论：只要用打印的方法，不论如何破解、如何发现新的突破方法，打出来的 PDF 文件膨胀、质量下降那是注定了的，想改都难，更何况还会受到软件方面的种种限制 ，所以奉劝各位还是趁早放弃。

另外一种所谓“利用中间 BMP”的转换方法也会产生问题：有人先用 BooX Viewer 或其它软件将 PDG 转换成 BMP，再用 Acrobat 或其它软件将 BMP 转换成 PDF。这种方法只能将 T1 图像无损转换成 PDF；对于 T2、T3，由于很难将 BMP 图像无损存入 PDF（那样尺寸膨胀太过厉害），只能再压缩成 JPG 或 JPEG 2000 后存入 PDF，因此质量下降、尺寸膨胀等问题是免不了的。

那么什么样的方法才是正确的转换方法呢？

我的回答是：条条大道通罗马，只要抛弃表层皮毛的束缚，直接深入到文件格式内部，就可以找到好的方法。就我自己来说，最常用的组合是 Pdg2Pic＋FreePic2Pdf：

先用 Pdg2Pic 将 PDG 直接解码成常规图像文件。能够将 PDG 转图像的软件不少，但是 Pdg2Pic 对除彩色/灰度 DjVu 外的图像都能无损转换，尤其是对多层 PDG 的无损分解，目前是独一无二的。

再用 FreePic2Pdf 将 Pdg2Pic 的结果合并成 PDF，黑白图像用缺省的 Jbig2 无损就好。

http://www.readfree.net/bbs/read.php?tid=183499&keyword=

PDG转图像、PDF的若干方法

作者：马健
邮箱：stronghorse@tom.com
主页：http://stronghorse.yeah.net
发布：2006.05.26

一、前言
二、截图法
三、打印法
四、BooX Viewer
五、pdg2bmp&jpg&tif&pdf&txt
六、Pdg2Pic
七、方法之比较与展望
八：题外话：图像文件转 PDF

一、前言

PDG 文件是超星公司电子图书的专有格式，需要用超星公司的专用浏览器才能阅读。本文讨论 PDG 转图像、PDF 的方法，仅出于研究目的，并无意对超星公司的版权进行任何形式的侵犯，也不希望任何人用本文讨论的工具或方法从事侵权活动。如果需要浏览 PDG 电子书，请通过购买点卡等方式，以合法的途径获得。

本文假定用户通过合法的手段获得 PDG 文件，只是由于希望能够在比超星浏览器更好、更方便的浏览器上阅读，并且不对转换出来的文件进行扩散的情况下，才需要将 PDG 文件转换成图像文件或 PDF 文件。

二、截图法

简单点说，就是通过截图的方法，直接将超星浏览器中显示的内容，截为图片，再将图片转换成 PDF 文件。

这个方法可能是世界上最简单、最朴素，也是最容易想到的方法，并且对于所有版本的超星浏览器和所有能够正常显示的 PDG 文件均适用。制约这个方法的因素包括：

页面大小超出显示区域，导致截图截不全。解决的办法包括：找一台支持高分辨率设置的 PC（现在 17"液晶已经很便宜，19"也快平民化了）；如果显卡支持旋转显示，则将整个屏幕旋转 90°显示，方便显示细长页面。
手工一页一页截图，劳动强度比较大。解决的办法就是用各种现成的按键、

鼠标录制/播放软件与屏幕截图软件相结合，或者自己做一个连翻页带截图的小软件，实现自动化操作。

截出来的图像可能需要进行整理，包括切边、图像文件格式转换等。

总之，截图发虽然有一些限制，用起来也比较麻烦，但很难被超星屏蔽，不失为一种终极的方法。

三、打印法

即在超星浏览器中发布打印命令，将正在浏览的 PDG 文件打印到 PDF 虚拟打印机（包括 Acrobat PDF 打印机、PDFFactory 打印机等），成为 PDF 文件。

这种方法也是较早被用于转换 PDG 文件的方法之一，而且用起来非常简单、方便，因此广为流传，导致后来超星阅读器针对这种方法加了一些限制，但是这些限制很快就被突破，然后双方就这样乐此不彼、义无反顾、周而复始地一轮、一轮折腾下去。虽然在无关的人看来有点无聊，但是投身其中的人经常都会为每一个微小的突破而激动 ，还真是有精神寄托的人生。

目前制约这个方法的因素包括：

超星浏览器对 PDF 打印机的封锁。 新版超星浏览器会检查打印机的名称，发现是 PDF 打印机则不让打印。不过超星软件毕竟没有人智能，打印机被人一改名就检测不出来了。也有人先将 PDG 打印到支持 PostScript（PS）文件格式的真实打印机，再用 Acrobat 将 PS 文件转换成 PDF 文件，以绕过超星对虚拟打印机的检查。

超星浏览器对打印页数的限制。超星浏览器会限制合法用户每个月的打印总页数，够数（每月一千页）后就不允许打印。解决的办法包括将 ssreader.ul 文件属性改为只读，或定期对这个文件进行备份、恢复。

超星浏览器对打印效果的限制。 新版本的超星浏览器可能对以前的限制与反限制游戏终于厌倦了，因此干脆在打印的时候降低打印质量，导致打印出来的 PDF 图像质量与原始 PDG 文件差很多。针对这一招，目前网上提出的解决办法包括将新版 DLL 文件替换为旧版 DLL，或提高打印机 DPI 设置等。

总之，在我看来，打印法虽然简单方便，打印黑白图像也问题不大，但是打印灰度/彩色图像会出现图像质量衰减或文件膨胀等问题，所以至少我自己不到不得已是不会用的。

四、BooX Viewer

BooX Viewer 是 Momotalo、ShunCox、dd321 等合作开发的一款轻量、绿色 PDG 浏览器，无需安装，单独一个 EXE 文件即可运行，并且能够直接读取 ZIP 文件中的 PDG 文件等，这些都比原版超星浏览器强，也导致了它的流行。

早期版本的 BooX Viewer 提供一个“转换到 DjVu”功能，该功能先将 PDG 文件转换成 BMP，再转换成 DjVu 文件。因此也有人利用此功能的前半部分，先将 PDG 文件转换成 BMP，再将 BMP 转换成 PDF。不过这个功能在后来的版本

中已经取消了，并且加了一些类似广告的限制。

BooX Viewer 的开发基于对 PDG 文件格式的分析，不需要超星浏览器或 DLL 的支持，并且能够解码加密的 10H 等格式，这些都让我对其开发者充满了敬意。

五、pdg2bmp&jpg&tif&pdf&txt

这个软件是 coolman 开发的，对 PDG 的支持（包括 OCR）基于超星 Pdg2 控件，对图像、PDF 的支持基于 Pegasus ImagXpress Professional 控件，运行前需要先注册控件。

这个软件的发行范围很窄，最新版是多少我也不知道，只能以我手上现有的 3.8b0419 版来说事。在使用这个版本的过程中，我发现它存在下列限制：

直接将 PDG 转换成 PDF，则所有彩色、灰度图像均变成黑白图像。解决的办法是先转换成 BMP，再用其它软件将 BMP 转换成 PDF。 但是不知道为什么，pdg2bmp&jpg&tif&pdf&txt 没有文件重新编号功能，所以在从 BMP 转换成 PDF 时，页面顺序调整起来很麻烦。

将 PDG 转换成 BMP 等图像格式时，允许使用多线程并行转换，但是似乎稳定性会随之下降，所以我都只敢用单线程转换。

最要命的一点就是：这个软件在转换时需要占用系统剪贴板，因此如果在转换过程中同时用 Office 等软件干活（没办法，转换过程实在是太漫长了），则复制/粘贴功能将失效。我先是在工作时发现了这个问题，然后用剪贴板监视软件证实了我的猜测。对剪贴板的占用不仅影响前台软件的正常使用，而且由于 Windows 本身对系统剪贴板的限制，在转换 幅面很大的 PDG 文件时会转不了。

虽然有一些问题，但是这个软件支持加密的 AAH 格式等（除该软件外，coolman 还开发了一些独立运行的 PDG 解密软件），这些都让我对 coolman 及其作品充满敬意。

六、Pdg2Pic

在发现 coolman 的 pdg2bmp&jpg&tif&pdf&txt 会占用系统剪贴板后，我 google 了一下，还真查到了一段源代码，虽然我不可能看到 pdg2bmp&jpg&tif&pdf&txt 的源代码，但我相信它的核心应该与这段代码相似。不过在多看了两遍这段代码后，我觉得既然已经用了 Pdg2 控件，为什么不用它提供的其它接口获取图像，干嘛非要用系统剪贴板？为了证实我的想法的可行性，我花了点时间写了 Pdg2Pic 这个软件，顺便对我在使用 pdg2bmp&jpg&tif&pdf&txt 过程中发现的一些问题做了改进，包括：

转换过程不占用系统剪贴板，不影响用户在前台的正常工作。

可以自动将文件按封面、前言、目录、正文、附录的顺序排列，也可以手动调整文件顺序。

提供预览功能，在转换前可以先浏览 PDG 图像。

PDG 文件的扫描 DPI 自动转存入生成的 TIFF、PNG 文件，便于在转换成

PDF 文件时设置页面大小。

如果检查发现 PDG 文件是纯正的 JPG 文件，将不进行任何转换，直接将 PDG 复制为 JPG；黑白 PDG 文件转存为采用 CCITT G4 压缩的 TIFF 文件，以获取高压缩比；灰度/彩色 PDG 重新压缩为有损的 JPG 或采用 JPEG 压缩的 TIFF 文件，或无损压缩的 PNG 文件，或 JPEG 2000（有损/无损）。

由于我没有时间对加密 PDG 文件进行研究，因此 Pdg2Pic 不像 pdg2bmp&jpg&tif&pdf&txt 那样支持众多加密 PDG 格式。如果在 Pdg2Pic 统计的文件类型中出现加密格式，需要用 1xhkillerfull、aahkiller 等进行解密，然后再用 Pdg2Pic 进行转换。如果您原意提供 PDG 文件解密算法或代码，欢迎与我联系。

七、方法之比较与展望

上面介绍了一些 PDG 转图像、PDF 的方法，说句实在话，我认为没有一种方法是完美的，多多少少都有点毛病。而且在我看来，对于一个真正的 PDG 转 PDF 软件，至少还要解决以下问题：

从 PDG 目录到 PDF 书签（Bookmark）的转换。现在有些 PDG 图书是带目录的，在超星浏览器中打开后，左侧会显示树状结构的目录，便于快速定位需要阅读的页面。这个与 PDF 中的书签很类似，但是现在似乎还没有一个软件能够在将 PDG 转换成 PDF 时，顺手将目录转换成书签。

将图书信息（bookinfo.dat）插入 PDF 文件，便于用 Adobe PDF Reader 的搜索（search）功能，在一大堆 PDF 文件中找到需要的书。bookinfo.dat 其实是一个标准 INI 文件，用文本记录了书籍的书名、作者等信息，如果作为一个文本页插入 PDF 文件尾，无疑将给搜索提供一些必要的信息。

支持透明背景。原始的黑白 PDG 文件本身可以按透明背景色显示，因此在超星浏览器中可以根据需要对背景色、前景色进行设置，便于长时间观看。相比之下，PDF 的白底黑字看起来就累多了。其实 PDF Reader 本身是支持对页面背景进行定义的，条件是 PDF 中的图像必须采用透明背景。如果图像本身敲死了一定要用白底，PDF Reader 也没有办法。

现在最后一个问题可以通过 FreePic2Pdf 1.01 版解决，第二个问题可以通过超星章节目录提取器（SSContent）部分解决，其它问题解决起来都有点难度，不知道有多少人原意去做？至少我自己是没打算要去做，但是我很期待看到其他高手能够解决这些问题，推出更好的 PDG 转 PDF 工具。

八：题外话：图像转 PDF

本文的题目叫《PDG 转图像、PDF 的若干方法》，但是前面讨论的某些方法，如截图法只能得到图像，不能直接得到 PDF 文件，因此自然还需要讨论一个问题：怎样将图像转换成 PDF 文件？

别人怎么想的我不知道，我自己认为比较好的转换方法有两种：

1、用 Adobe Acrobat Professional 的 Create PDF from Multiple Files，而不用它的虚拟打印机

这种方法的优点是：

如果在转换前先指定黑白图像用无损 JBIG2 压缩，可以获取最高压缩比。

可以获得经过线性优化的 PDF 文件，这种文件在通过网络浏览时可以边浏览边下载，因此也被称为 Fast Web View 文件。但是对于只在本地阅读的 PDF 文件来说，我认为这种优化只会增加文件长度，不会节省实际的打开时间。

这种方法的缺点是：

对于灰度/彩色图像，可能会因为重新采样压缩而造成图像质量衰减或文件膨胀。这方面的讨论参见我写的《图像转 PDF 的问题、方法及题外话》。

如果一次需要处理几本书，操作起来有点麻烦。

如果图像大小不一，转换出来的页面大小也不一致，看起来有点心烦。

至尽为止，我还没有找到如何设置，才能在转换黑白图像时，能够将背景设置为透明。如果您知道，还请不吝赐教。

2、用 FreePic2Pdf

这种方法的优点是：

按照缺省设置，黑白图像转换成 CCITT G4 数据流，JPEG/JPEG 2000 数据流直接嵌入 PDF 文件，不会因为重新采样压缩而造成图像质量衰减或文件膨胀。

便于批量处理，包括设置页面大小、页边距，在开始转换前调整文件顺序也很方便。

从 1.01 版开始，对于黑白图像，可以自动转换成透明背景色。由于有了这个功能，我甚至打算在有了好的 PDF 转图像软件后，把以前收集的一些扫描版 PDF 还原成图像，再用它转成 PDF。原因无它， 白底黑字的 PDF 实在是看怕了。

最重要的一点：它是免费的绿色软件，个人使用不存在法律后患。

这种方法的缺点是：

由于缺乏相关开源项目的支持，因此不支持 JBIG2 压缩，只能采用 CCITT G4 压缩黑白图像，转出来的 PDF 文件可能会比 Acrobat 用 JBIG2 转出来的大一点。如果您手上有没有法律问题的 JBIG2 压缩源代码，欢迎与我联系。

没有线性优化功能。如果您制作的 PDF 只在本地阅读，不打算通过 IE 在线阅读，这个缺点将变成优点。

总之，现在也没有十全十美的图像转 PDF 软件，也许这样的方法会是更好的选择：转换还是用支持 JBIG2 和 JPEG 2000 的 Acrobat 转，但是做一个小程序，将它转出来的 PDF 文件的黑白图像的背景改为透明。由于是单纯的字符替换，所以软件很好写，并且不需要其它第三方代码或控件的支持。

http://www.readfree.net/bbs/read.php?tid=305551&keyword=

说“DPI”

作者：马健
邮箱：stronghorse@tom.com
主页：http://stronghorse.yeah.net
发布：2007.03.08
更新：2007.04.02

目录

一、基本概念
二、图像文件中的 DPI
三、PDG 文件中的 DPI
四、PDF 文件中的 DPI
五、DjVu 文件中的 DPI

一、基本概念

DPI 是 Dot Per Inch 的缩写，字面意思就是“每英寸点数”，即在一英寸的长度上，设备能够显示、打印、扫描、拍摄……多少个点，其基本计算公式为：

DPI＝象素点数÷英制长度（点/英寸）

习惯上，设备的象素点阵坐标系称为物理坐标系，其它坐标系称为逻辑坐标系。因此本文把点阵图像的象素尺寸称为物理尺寸，英制/公制尺寸称为逻辑尺寸；而 DPI 则可以看作是在物理尺寸和逻辑尺寸之间进行转换的桥梁：

・在生成（扫描、拍摄）图像时，图像尺寸和设备 DPI 决定了最终图像的点阵尺寸。如一张宽度为 5 英寸的图片，在 300 DPI 的扫描仪上扫描，最终点阵宽度＝5×300＝1500 象素。
・在输出（显示、打印）图像时，图像象素尺寸和设备 DPI 决定了最终图像的尺寸。如点阵宽度为 1500 象素的图像，在 600 DPI 的打印机上输出，最终图像宽度＝1500÷600＝2.5 英寸。

从上面举的例子可以看出，当输入、输出设备的 DPI 不一致（这种情况很常见）时，可能会导致输出图像大小与原始输入图像大小不一致（上面例子中差了一倍）。为了解决这种不一致，让图像在所有输出设备上看起来都一样大小，通常作法是先按输入设备的 DPI 将点阵宽度换算回图像的原始逻辑宽度（英制、公制宽度），再按输出设备的 DPI 将逻辑宽度换算成输出点阵宽度。

还是以上面举的例子来说，5英寸宽的图像在300 DPI扫描仪上扫描后点阵宽度为1500象素，为了在600 DPI的打印机上打印这张图片时还能打印成5英寸宽，就需要先对这张图片进行放大，在将它放大一倍，宽度达到3000象素后，在600 DPI的设备上打印宽度＝3000÷600＝5英寸。

打印时的这种缩放，当然会对最终打印出来的图像造成影响。这就是为什么在用虚拟打印机将图像文件打印成PDF或DjVu时，最终文件大小、质量往往会与虚拟打印机DPI有关的原因。

当然除了使用英制单位外，DPI也可能使用公制单位，这时它的单位通常是“点/米”，即1米长度上有多少个象素点。从习惯上考虑，本文对用公制单位表示的DPI还是称为DPI，而不是DPM（Dot Per Meter）。

二、图像文件中的DPI

由于很多图像浏览、处理、印刷软件都会用输入设备的 DPI 计算原始图像逻辑尺寸，所以大多数常见图像格式都允许在文件结构中记录图像生成时的设备DPI：

・BMP 格式：在 BITMAPINFOHEADER 结构体的 biXPelsPerMeter、biYPelsPerMeter字段中定义， 从字面上看采用的是公制单位。
・PNG格式：在png_info结构体的x_pixels_per_unit、y_pixels_per_unit字段定义，单位是公制还是英制由phys_unit_type描述
・TIFF格式：通过TIFFTAG_XRESOLUTION、TIFFTAG_YRESOLUTION这两个tag定义，单位通过TIFFTAG_RESOLUTIONUNIT定义。
・JPG格式：JFIF格式的在APP0段（FF E0）的X_density、Y_density字段定义，单位由density_unit描述；EXIF格式的在Xresolution、Yresolution中定义，单位由ResolutionUnit描述。

从上面对图像格式的描述可以看出，大多数图像格式在存储 DPI 时都有两个共同特点：

1、支持公制和英制单位。这大概就是所谓的“国际化”支持。
2、支持x、y方向各设置不同的DPI。这是由设备特性决定的：由于物理限制，某些扫描、印刷设备在两个方向上的运动精度可能不同，DPI当然也不一样，如我碰到过一台印刷设备的 DPI 就是 600×300。在 libtiff 提供的测试图像中，有两张CCITT编码的Fax图像的x、y向DPI也不同，显示时如果不注意，就可能显示出变形 （园变成了扁椭圆）的图像。

三、PDG文件中的DPI

在目前发布的V1、V2版PDG文件结构中，并没有任何关于DPI的描述，那么超星浏览器所带的pdg2控件的GetDpi方法返回的DPI又是怎么回事？

其实说穿了很简单：GetDpi 返回的 DPI 值是动态算出来的——如果图像宽度大于 1200 象素，则认为是 300 DPI；否则是 150 DPI。详见网上读书园地论坛（http://www.readfree.net/bbs/index.php）里 cheming 先生的相关 分析文章。

对于通常的书籍页面来说，扫描图像宽度达到 1200 象素以上已经足够清晰，所以通常 PDG 收藏者（这么说是不是有点无聊？^_^）也用 DPI 值作为区别清晰版、快速版的标准：如果 Pdg2Pic 报告 300 DPI，则认为是清晰版；否则认为是快速版。

大多数情况下，这种判别标准应该说是准确的，不过偶尔也有失手的时候：有时图像实在太大，即使已经按超星快速版的质量标准进行了压缩（长、宽各砍掉一半），但宽度还是超过 1200 象素，这时就会被误判为清晰版。不过这种情况毕竟少见，我也只在读书园地见过一页。

快速版 PDG 的数据流格式通常是 DjVu，而 DjVu 格式要求在 INFO 段中填写 DPI 值。从实际解出来的数据流看，PDG 中的 DjVu 有填 150 DPI 的，也有填 100 DPI 的。至于 100 DPI 是实际扫瞄时的 DPI，还是超星程序员偷懒使用了 djvulibre 的缺省值，那就不知道了。

从目前了解的情况看，PDG 文件的 DPI 对显示没有任何影响，超星浏览器显示时都是按照图像实际象素尺寸进行计算。不过 Pdg2Pic 在将 PDG 文件转换成图像文件时，会按照通常图像处理软件的习惯，将计算出来的 PDG 文件 DPI（150/300）写入转换出来的图像文件。 从目前的情况看，这种做法似乎给某些原先习惯使用“打印大法”，现在转为使用 FreePic2Pdf 的人造成了困扰，详见下一节说明。

四、PDF 文件中的 DPI

PDF 文件中同样没有任何地方存储 DPI，但是所有 PDF 生成、处理软件都绕不开 DPI：PDF 文件格式规范规定，在描述页面及页面中的对象时，所有表示大小、位置的数值必须折算成“用户单位（User Unit）”。用户单位的缺省值为 1/72 英寸，即 PDF 文件的页面 DPI 为 72。当然这个值也可以在文件中加以更改，不过很少有人这么干。

以一幅宽度为 1500 象素，扫描 DPI 为 300 的图像为例，如果要在 PDF 中看起来的大小与被扫描的原始图像大小一样（5 英寸），则在 PDF 页面描述中，必须定义这幅图像的宽度＝1500÷300×72＝360（用户单位）。

而如果要使图像上每个象素点用屏幕上一个象素点显示，则应该在 PDF 页面描述中，定义这幅图像的宽度＝1500÷96×72＝1125（用户单位）。其中 96 是屏幕的 DPI 值。

理论上说，对于图像对象，PDF 允许将图像的物理表示与逻辑表示分开（参见我写的《图像转 PDF 的问题、方法及题外话》）。还是以上面这幅图像为例，PDF 制作软件可以将原始图像按照 1500 象素宽度存储为图像对象，然后在页面中对这个图像对象进行引用，并按用户单位定义这个图像在页面上的位置、大小。在这种情况下，改变显示时的位置、大小，并不会对原始图像数据造成任何影响。FreePic2Pdf 就是这么干的，所以在 FreePic2Pdf 中改变 DPI，并不会改变最终 PDF

文件的大小，因为仅仅是图像的位置、大小被重新计算，图像数据流不会变。

但是对基于虚拟打印原理的转换软件来说，就是另外一回事了，参见前面“基本概念”部分。在这些软件中，改变目标 DPI 值（虚拟打印机的 DPI），将对最终 PDF 的文件大小、清晰度产生影响，因为软件会按照虚拟 DPI 重新缩放图像本身。这也是我为什么一直对“打印大法”比较排斥的原因：原始图像可能会在打印过程中被重新采样。

另外对于 PDF 中的黑白图像来说，如果选择 JBig2（从 PDF 1.4，即 Acrobat 5.0 开始支持）有损压缩，则结果可能受 DPI 设置影响，也可能不受，由具体的符号匹配（Symbol Match）代码决定。《PDF Reference fifth edition》第 3.3.6 节对 JBig2 压缩的原理及作用描述如下：

The JBIG2 encoder builds a table of unique symbol bitmaps found in the image, and other symbols found later in the image are matched against the table. Matching symbols are replaced by an index into the table, and symbols that fail to match are added to the table. The table itself is compressed using other means. This method results in high compression ratios for documents in which the same symbol is repeated often, as is typical for images created by scanning text pages. It also results in high compression of white space in the image, which does not need to be encoded because it contains no symbols.

这段话对 JBig2 进行了专家级概括。而在判断符号是否匹配时，某些软件会结合 DPI 进行判断（参见后面“DjVu 文件中的 DPI”中对有损 JB2 的描述，JBig2 和 JB2 不仅名字像，原理也一样）。但是在 FreePic2Pdf 中没有做这种结合，所以即使改变 DPI，对 FreePic2Pdf 中的有损 JBig2 压缩结果也没有什么影响。

虽然如前所述，在 FreePic2Pdf 里改变 DPI，不会改变最终图像数据流的长度，但是由于 PDF 显示软件本身的原因，在 FreePic2Pdf 里改变 DPI 后，还是可能对最终显示效果造成影响。例如某清晰版图像宽度为 1634，按照缺省的 96 DPI 转换，在 1024×768 的屏幕上显示时选择“适合宽度”，在工具条上显示实际缩放比例为 59%，即“缩小”；而如果按照 300 DPI 转换，显示比例将为 185%，即“放大”。某些软件在显示 PDF 时，会先将图像解码，然后缩放到逻辑尺寸，再按照用户选择的缩放比例缩放成显示尺寸。对于这种软件，最终“缩小”的效果要好于最终“放大”的效果。当然最新的显示软件都没这么笨了，一般不经过中间环节，直接将图像从原始大小缩放到最终大小，这时 DPI 的作用就没这么明显。

正因为这个原因，FreePic2Pdf 里缺省 DPI 是屏幕 DPI——96 DPI，以免在某些软件上显示时造成质量下降。

WinDjView 也有类似的毛病，所以在 FreePic2Pdf 之后开发的 DjVuToy 里，我本不想再让用户自己设置 DPI 的，但总有那么几个人认为不能自己设置就会浑身不舒服，所以最终还是 把这个功能开放了。

五、DjVu 文件中的 DPI

在文件存储方面，按照 Lizardtech 公司出版的《Lizardtech DjVu Reference DjVu v3》第 8.3.11 节规定，DjVu 文件在 INFO 段中定义 DPI，单位为英制。与其他常见图像格式不同，DjVu 中没有区别 x、y 方向 DPI，即认为两个方向的 DPI 是一样的。在将 x、y 方向 DPI 不同的图像转换成 DjVu 时，必须对此加以

注意，以免转完后变形。

在文件转换方面，如果按照 djvulibre 源代码实现，DPI 仅对采用有损 JB2 压缩的前景蒙板层有影响，对无损 JB2 压缩的前景蒙板层，及前景层、背景层无影响（对 DjVu 各层的说明见我写的《说“层”》，JB2 原理见前面“PDF 文件中的 DPI”）：

•在 JB2 有损压缩时，一般要在开始搜索符号（字符）前，先对版面进行清洁（clean），即将细小的孤立点当作噪声点去除。具体多大的点算“细小”，则由 DPI 决定。djvulibre 中 cjb2.cpp 的相关计算代码为：

```
dpi = MAX(200, MIN(900, dpi));
largesize = MIN( 500, MAX(64, dpi));
smallsize = MAX(2, dpi/150);
tinysize = MAX(0, dpi*dpi/20000 - 1);
```

即在计算时，先对用户指定的 DPI 值有效性进行检查，小于 200 DPI 的一律算 200 DPI，大于 900 DPI 的则算 900 DPI；largesize 是符号最大尺寸，长、宽大于等于此值的连通域均会被拆分，以避免符号粘连；smallsize 是符号最小尺寸，长、宽小于等于此值的连通域均 将视需要合并，以避免一个字母被拆成两半（如字母 i 上面的点和下面的竖线通常是不连通的）；tinysize 是噪声点尺寸，长、宽小于等于此值的孤立连通域 可能会被当作噪声点加以清除。

简单来说，对于同一个原始图像，如果 DPI 小于等于 200，只有长、宽为 1 个象素的孤立点会被 clean 掉；在 DPI 为 300 时，长、宽小于等于 3 个象素的孤立区域会被 clean 掉。

•如果 JB2 选项选的是“去斑（clean）”，通常在完成上述去斑过程后，即按无损压缩处理；如果选择的是“有损（lossy）”，则在去斑后，再进行符号匹配，并对“相同”（其实是“相似”）的符号进行合并。至于什么样的符号算“相同”，则与 DPI、有损级别（losslevel）有关：DPI 和 losslevel 值越大，允许两个“相同”符号之间的差异越大，即越容易将两个有差异的符号判别为“相同”。在 djvulibre 中，losslevel 的合法值是 0～200：0 即“无损”，1 即“去斑”，通常“有损”时 losslevel＝100。

djvulibre 的符号匹配算法是外国人写的，对字母、数字来说，有损 JB2 压缩能在较小的风险代价下，获得较高压缩比。但是对中文来说，故事就没这么美好：很多字本来就很像（如“己”和“已”、“人”和“入”、“面”和“而”等），扫瞄质量如果差点（笔画缺胳膊少腿），再被 clean 过程去掉点孤立的笔画或点，在有损压缩时就 可能将相似字判别成相同字。在读书园地 BBS 上已经有人贴过这样的实例：采用 CCITT 无损压缩的 02H 的 PDG 文件上没有错别字，但是同一个文件的 05H 版（DjVu）就出现错别字（将“面”和“而”混了）。

当然上面说的是 djvulibre 的实现，超星用它是确定无疑的，其他 DjVu 制作软件，尤其是各大商业 DjVu 制作软件是否也这样处理就不知道了。

在文件显示方面，在 WinDjView 中，按照不同的显示比例，决定是否按照 DPI 折算图像宽度：

•如果选择“实际尺寸（Actual Size）”，则按照 DjVu 图像的实际象素尺寸显示，即图像上的一个象素点占用屏幕上的一个实际象素点。

•选择“适合宽度（Fit Width）”、“适合高度（Fit Height）”、“适合页面（Fit Page）”、“拉伸（Stretch）”时，与选择“实际尺寸”一样，WinDjView 根本就不

管 DjVu 文件里 DPI 是怎么设置的，直接将象素点阵缩放到窗口宽度、高度显示。

•如果选择“100％”，则按照图像 DPI，将图像点阵尺寸按象素折算成英寸，然后按照屏幕 DPI，折算成屏幕点阵尺寸显示出来。

• 选择其他百分比时，同样是先将图像点阵尺寸按 DPI 折算成英寸，然后按比例缩放，再按照屏幕 DPI，折算成屏幕尺寸显示出来。所以有些高 DPI 的图像， 放大后看起来效果会有点差。这有点类似前面说的某些 PDF 显示软件。

正因为以上原因，我自己在看 DjVu 文件时，尽量不选按百分比显示，都是“Fit Width”、“Fit Page”或“Actual Size”。

在将 PDG 转换成 DjVu 时，DjVuToy 直接对图像数据流进行操作，没有任何无谓的缩放，所以在选择无损 JB2 的情况下，就算改变 DPI 值，也只不过改变了最终 DjVu 文件的一个字段值而已，不会造成文件大小的变化。

对于其他基于虚拟打印原理实现的 DjVu 转换工具，与 PDF 虚拟打印一样，DPI 是个绕不过去的坎：打印机需要按照 DPI，计算最终图像大小。为了与这个最终尺寸相匹配，需要对图像进行缩放（通常是放大，因为打印机分辨率比屏幕高），这也是为什么在使用这些软件时，设置不同的 DPI 不仅会改变文件大小，而且会改变最终图像质量的原因 之一。

而在 DjVuToy 的“文件宽度”功能里，则通过更改每一页 INFO 段中的 DPI 值，保证在 WinDjView 中按照 100%显示时，所有页面宽度一致。例如想将页面宽度统一为 5 英寸，则先找出各页的象素宽度，然后设置每一页的 DPI＝该页象素宽度÷5 即可。这个也不需要更改原始图像数据。当然由于 DjVu 的 DPI 必须是整数，四舍五入后显示宽度可能会有一点微小的差异，影响不大。

http://www.readfree.net/bbs/read.php?tid=186451&keyword=

作者：cheming

PDG文件的DPI

看了一下 PDG2.DLL，知道 DPI 信息不是保存在 PDG 中，而是计算出来的

```
.text:1001E220 ; __int32 __cdecl IT_Pdg01__dpi_get(void *pThis)
.text:1001E220 IT_Pdg01__dpi_get proc near ; DATA XREF: .rdata:100D4E3C├o
.text:1001E220 ; .rdata:100D54C4├o
.text:1001E220
.text:1001E220 pThis = dword ptr 4
.text:1001E220 arg_4 = dword ptr 8
.text:1001E220
.text:1001E220 mov eax, [esp+pThis]
.text:1001E224 lea ecx, [eax+0C0h]
.text:1001E22A call GetDPI
.text:1001E22F mov ecx, [esp+arg_4]
.text:1001E233 mov [ecx], eax
.text:1001E235 xor eax, eax
.text:1001E237 retn 8
.text:1001E237 IT_Pdg01__dpi_get endp
.text:10011B50 GetDPI proc near ; CODE XREF: IT_Pdg01__dpi_get+A├p
.text:10011B50 mov edx, [ecx+2194h]
.text:10011B56 xor eax, eax
.text:10011B58 cmp edx, 1200
.text:10011B5E setle al
.text:10011B61 dec eax
.text:10011B62 and eax, 150
.text:10011B67 add eax, 150
.text:10011B6C retn
.text:10011B6C GetDPI endp
```

根据上述程序可知：

pdgWidth>1200 : pdgDPI = 300
else: pdgDPI = 150

http://www.readfree.net/bbs/read.php?tid=277196&keyword=

说“层”

作者：马健
邮箱：stronghorse@tom.com
主页：http://stronghorse.yeah.net
发布日期：2007.01.10
最近更新：2007.02.20

目录

一、PDG 中的层
二、PDF 中的层
三、DjVu 中的层

声明：本文仅为笔记的一部分，最多算备忘录，如果读的时候觉得无头无尾请不必惊讶。

一、PDG 中的层

早期 V1 版 PDG 文件结构中并没有层的描述字段，因此也没有分层不分层的说法：如果书籍页面上既有文字又有插图，则连文字带插图存储成一个大 JPG。如果插图在页面上只占用很少一点面积，这样的存储方式当然很不经济：JPG 并不适合存储黑白文字 。而如果能将文字与插图剥离开，文字用超星惯用的 CCITT 压缩存储，其压缩比要比 JPG 高很多，显示效果也会好很多。

这种文字与插图相区别的定义，从 V2 版 PDG 文件结构开始出现：在主文件头中除主文字层的宽、高、数据长度等，还定义了插图的个数；在主文件头后顺序编列各插图的描述信息，包括：插图左上角点位置、插图高度、插图宽度、插图数据起始位置（相对文件起始位置的偏移量）、插图数据长度（字节数）等。

浏览器在绘制这种 PDG 时，先读取文字层数据（CCITT 或黑白 DjVu 数据流），在背景上画出文字，然后逐个读取插图数据，读一个画一个。即从底往上，依次为背景（超星允许设置背景图案或背景颜色）层、文字层、第一张插图、第二张插图、……、第 n 张插图。

PDG 所有插图均为矩形，数据流格式为 24 位 JPG 或 DjVu，偶尔也会将 JPG 或 DjVu 数据流封装成一个完整的 PDG 文件作为插图数据，这样就产生了所谓的“嵌套 PDG”：在一个 PDG 文件里，还包含一个或几个 子 PDG，每个子 PDG 都是一个插图。

理论上 PDG 文件的插图之间允许出现重叠，重叠时上层插图覆盖下层插图的重叠部分。但是在实际的 PDG 文件中，尚未见到重叠的情况，感觉超星找到了很好的方法，能够保证插图切割后的总面积最小，没有任何浪费。

二、PDF 中的层

PDF 中的层稍微复杂一些，我将其区分为“狭义层”和“广义层”。

所谓“狭义层”，指的是在 Adobe 出版发行的《PDF Reference fifth edition》第 4.10 节中定义的 Optional Content（可选内容）。我称呼它为“狭义层”，是因为在 Adobe Acrobat 7.0 的界面、帮助里，都用“层（layer）”这个词汇来明确称呼它。后面叙述的“广义层”就没有享受这样的待遇。

用 Optional Content 定义的层，类似于 Photoshop 中“层”的概念，事实上 Acrobat 界面左侧“层”窗口（排在“书签”窗口下面）的操作，也和 Photoshop 里的层操作差不多：点击眼睛图标，即可显示和隐藏指定的层。在同一个层上可以放置多个对象 （文本、图像、按钮等），隐藏、显示某层时，该层所有对象一起隐藏、显示。判断一个 PDF 文件是否采用 Optional Content 来定义层的方法也很简单：用 Acrobat 打开 PDF 文件，如果点开“层”窗口能够看到内容，则该 PDF 肯定包含层，否则不包含。

我的同事在设计 PDF 表单时，喜欢在表单里用层来定义一些装饰性或提示性的元素，这些元素在打印时会隐藏。不过这几个家伙以前是做网页的，所以我一直怀疑他们是在 HTML 里用 DIV 用惯了，转到 PDF 来还是改不了老习惯:)，过一段时间自然会改用更简单的方法。我也曾在国学数典 BBS 上见 到有人用 Adobe InDesign，做了一本用 Optional Content 定义的多层 PDF：一层是扫描图像层，一层是文字层，还有一个附加层上放置了层选择按钮：点击文字层选择按钮，则隐藏图像层、显示文字层；点击图像层按钮则反之。不过可能这种书籍制作要求实在太高，用处又不大，我到现在也只见过这么一本。

通常大家说的“双层 PDF”，其实和用 Optional Content 定义的层一点关系都没有，所以用 Acrobat 打开以后，点“层”窗口什么也看不到，即这种“双层 PDF”的“层”，其实不是 Acrobat 所认可的“层”，只不过大家都这么说，而且从逻辑上来说确实有点“层”的意思，所以我管这个叫“广义层”。

这种广义的“双层 PDF”，底层是扫描图像层，上层是从扫描图像 OCR 出来的文字层，只不过文字设置为透明（在 Foxit PDF Editor 的“文本属性”页中，“文本模式”为“没有填充和空心的文本（不可见）”，正常文字应为“填充文本”或其他）。这样通过 PDF 浏览器阅读的时候，看到的是底层原汁原味的扫描图像，各种公式、图表都和纸上看到的一样，但是“搜索”或用文字选取工具选取时，又可以直接对上层文字进行操作。因此这种 PDF 能够较好地避免纯扫描版不能检索，纯文字版排公式、表格困难的问题；同时兼有扫描版保持原文版式，纯文字版可以检索、复制/粘贴的优点。Adobe Acrobat 8 就把能制作、转换这种双层 PDF 作为一大卖点加以介绍，虽然它的中文 OCR 引擎实在不怎么样。

这种双层 PDF 为了保证用文字选取工具选择文字时能够准确定位，通常要求严格实现“字压图”，即上层透明文字，要与底层图像上对应文字的大小、位置完全一样（当然真正完全一样是不可能的，只能尽量对准）。而版面上文字的间距、大小、字体可能变化多端，所以在生成 PDF 的时候，通常对每个字的位置进行单独描述，这就导致生成的结果基本上不可再校对、编辑。不信可以用 Foxit PDF Editor 打开一个生成好的双层 PDF，用鼠标一个一个点过去，看是不是一次只能选一个中文字或一个字母、数字？对于这样的东西，我想就算是《大话西游》里的唐僧转世，校对上几页也会疯掉。

当然如果非校对不可，也不是没有办法，而且办法还不止一种。

如果错误不多，可以采用下面的方法：

1、用 Acrobat 或其他工具，将 PDF 中的文字信息全部导出成文本文件。

2、用 Foxit PDF Editor 打开 PDF。

3、在文本文件中看到错别字时，在 Foxit PDF Editor 里选中对应的字进行修改。

4、改后别忘了存盘。

如果看了文本后感觉错误很多，则可以用下面的方法：

1、用 Foxit PDF Editor 打开 PDF，进入需要修改的页面。

2、先按 Ctrl+A 选择全部对象，然后按住 Ctrl 键，在页面空白处点一下鼠标左键，将背景图排除在选择之外，以确保选中的都是文字对象。

3、在“文本属性”页中，将“文本模式”从“没有填充和空心的文本（不可见）”改成“填充文本”，然后点“更改”。

4、现在看到文字显示出来了吧？不过缺省文字颜色一般是黑色，看起来费劲，可以保持文字选中状态，进入“填充颜色”页，改变颜色、透明度，然后点“更改”，将文字压缩和背景颜色区别开。

5、现在就可以一个字、一个字地选中、修改了。改完后再按照步骤 2 选中全部文字，按照步骤 3 将“文本模式”改回“没有填充和空心的文本（不可见）”，存盘即可。

FreePic2Pdf 在将多层 PDG 转换成 PDF 时，用的是与 PDG 浏览器绘制多层 PDG 时相同的顺序：在一个空白页面上，先放上底层文字层，然后将插图顺次堆叠上去，插图的位置由 Pdg2Pic 在生成的接口文件时加以描述。通常 PDF 文件中处理图文混排都用这种模式。

三、DjVu 中的层

DjVu 中层的含义比较明确，在文件格式规范中有清晰的定义：分为背景层和前景层，但是前景层又可以分解成前景蒙板层和前景层，所以一个标准的多层 DjVu 其实可以包含三层：背景层（background layer）、前景层（foreground layer）、前景蒙板层（foreground mask）。在 Lizardtech 公司出版发行的《Lizardtech DjVu Reference DjVu V3》中，对他们的说明是：

・The background layer is used for encoding the pictures and the paper texture.

・The foreground layer is used for encoding the text and the drawings.

・The main component of the foreground layer is a bi-level image named the foreground mask. The pixel size of the foreground mask is equal to the size of the DjVu image. It contains a black-on-white representation of the text and the drawings.

我自己的经验和理解：

・前景蒙板层：对页面上具有清晰轮廓的对象加以描述，如文字、表格、线框、线条等。

・前景层：为前景蒙板层所描述的轮廓提供填充色。

・背景层：包括背景、底纹、插图等。

在显示 DjVu 页面的时候，实际象素点的颜色根据蒙板层中对应点的颜色进行选取：如果蒙板层中该点为黑，则该点实际颜色从前景层取，否则从背景层取。

如果希望对 DjVu 的三层结构进行详细研究，可以用 DjVuToy v0.04+，导出 DjVu 的文件结构和三层图像，与《Lizardtech DjVu Reference DjVu V3》中的说明进行对比。

我个人一直认为，DjVu 所吹嘘的清晰度、压缩比，都与它的分层方式有很大关系。

先说清晰度。由于采用模板层对轮廓进行选取，最终显示出来的 DjVu 图像在文字、线条等轮廓边缘显得非常锐利、清晰，不会出现 JPG 图像那样的细小碎片（高频杂波），看起来自然舒服。

再说压缩比。对于文字，DjVu 在前景蒙板中使用 JB2 压缩，这种压缩与 PDF 中的 JBIG2 相似（在开源项目 djvulibre 中对二者的渊源有过描述），对于黑白图像，尤其是以字母、数字为主的黑白图像，具有非常高的压缩率。而对于插图、底纹等彩色信息，DjVu 通常在背景层中用 IW44 压缩。IW44 与 JPEG 2000 也颇有渊源（这在 djvulibre 里也有描述）。

总之，DjVu 通过将页面内容分解，针对不同的内容，在不同的层中用不同的算法进行压缩，其还原效果、压缩比自然要比其他只有一层的图像格式，如 JPG、JPEG 2000 要好。

另外我相信，如果将 DjVu 的分层技术活用到 PDF 格式上，也可以减小扫瞄版 PDF 文件的体积：目前的转换软件在将图文混排的扫瞄页面转换成 PDF 时，采用的是早期 PDG 的做法，即将整页作为一个图像存入 PDF，这样当然比较缺乏效率。如果能够向后来的 PDG 学习，将文字区域和插图区域分开，分别采用不同的压缩算法（PDF 支持不同的图像对象采用不同的压缩算法），则整个 PDF 文件的长度可以大大减少。目前只是没有人去研究这种分割的方法，并加以实现。

另外在 DjVu 中也支持类似“双层 PDF”的东西：看到的是一张扫描出来的图像，用鼠标一选则能选中文字，或对文字进行检索。这种“双层 DjVu”的文字部分在 DjVu 文件格式规范中同样没有使用“层（layer）”这个词 加以定义，而是叫做“Text Chunk”。和 PDF 一样，这种 DjVu 也要求“字压图”，因此其结果也基本上不可再校对、编辑：不是技术上不可以，是人的精力和精神无法承受。

http://www.readfree.net/bbs/read.php?tid=176715&keyword=

图像转**PDF**的问题、方法及题外话

作者：马健
邮箱：stronghorse@tom.com
主页：http://stronghorse.yeah.net
发布：2006.04.05
更新：2006.11.21
测试用例：用例 1（一组各种格式的图片），用例 2（libtiff 3.7.1 所带测试图片）

友情提示：虽然我已经尽量简化，但是这篇文章感觉还是有点绕，我自己都没有自信一遍就能看懂。如果您的时间很宝贵， 建议不要随便浪费，还是去做该做的事吧，这里讨论的不过是些鸡毛蒜皮的小问题。

目录

一、需要解决的问题
1、图像数据流重新压缩造成的问题
2、阅读的顺畅性问题
3、对特殊图像格式的支持问题
二、预备知识
1、PDF 支持的图像格式
2、图像在 PDF 中的物理表示
3、图像在 PDF 中的逻辑表示
三、问题的解决办法
四、小结
五、题外话一：PDF 转图像
六、题外话二：除了 PDF，还有什么？
1、多页 TIFF
2、JBIG2
3、DjVu
4、双层 PDF

一、需要解决的问题

图像转 PDF 似乎正在成为一个热门话题：对企业或组织来说，随着信息化的深入，需要将大量纸质档案电子化，实现在线查询、共享；对个人来说，随着家用数码相机等的普及，越来越多的人希望将电子图像信息转换成方便浏览、共享的格式。由于 PDF 文件本身的标准化、方便性，目前在企业和家庭应用越来越多，由此也带动了诸多图像转 PDF 软件的诞生。 当然“树林子大了，什么样的鸟都有”，至于鸟叫得怎么样，只能实际比较一下再说。

正好由于工作需要，我近期参与了某金融企业的无纸化办公平台建设项目，其中一个重要内容就是将该企业遍布全国的三十余个分支机构积存的巨额（最先开始扫描的一个分支机构就有近三百万页）纸质档案扫描成电子文档。作为项目负责人，我代表客户与国内数家扫描外包公司进行了接触、考察，并对能够搜集到的图像转 PDF 工具进行了测试、比较，在这个过程中我发现目前图像转 PDF 软件大致可以分为两类：

基于虚拟打印原理。最著名的大概要算 Adobe Acrobat Professional、PDF Factory。基于虚拟打印原理的软件开发门槛稍高一些（需要提供打印驱动程序），所以多为收费软件，通用性较好，除图像文件外还能将 Word 等所有可打印格式转换成 PDF。

直接将图像嵌入 PDF 文件。如 ACD Systems 出品的 ACDSEE 中的 Create PDF Wizzard、verypdf 出品的 Image2Pdf 等。直接将图像嵌入 PDF 文件的软件实现相对简单，所以收费、免费的都有。

从测试的结果看，我认为这两类工具普遍存在一些共性问题，包括图像数据流重新压缩造成的问题、生成的 PDF 文件的阅读顺畅性问题、对特殊图像格式的支持问题等。本文将对这些问题的成因、解决方法加以探讨。

1、图像数据流重新压缩造成的问题

对基于虚拟打印原理实现的转换软件来说，其工作过程为：

转换工具提供一个虚拟打印机。如 Acrobat 提供的打印机名为 Adobe PDF。

图像浏览软件打开图像文件，在收到打印命令后，象在真实打印机上打印一样，将图像逐象素描绘到虚拟“纸”上，形成发送给虚拟打印机的数据流。

虚拟打印机收到数据流后，根据图像的色彩空间等信息，选择“合适”的压缩算法，对数据流再次进行压缩以减小文件长度，然后将压缩后的数据流存入 PDF。

为了测试虚拟打印机对图像的处理，我选择了一组图像（用例 1），在 ACDSEE 8.0 Build 39 中打开，选中所有图像，然后选择“File->Print”，打印到 Acrobat 7.07 提供的 PDF 虚拟打印机（ACDSEE 和 PDF 打印机的所有参数均为默认值），然后比较原始图像数据和 PDF 中的图像数据，结果见表 1.1。表 1.1 中各种“解码器”的解释见本文后续的“PDF 支持的图像格式”部分，“PDF 中的图像数据”各栏中的数据来自开源的 PdfView。如果您有兴趣查看 PDF 文件内部细节，建议用 UltraEdit-32，仅看 PDF 文件结构 用 PdfView 足矣。

表 1.1 从 ACDSEE 打印图像到 Acrobat PDF 虚拟打印的结果

原始图像 PDF 中的图像数据

序号 说明 宽×长

（象素） 图像解码器 文件长度

（字节） PDF 解码器 BitsPerComponent

/ColorSpace 数据流长度

（字节）

01 黑白 TIFF 1728×1103 CCITT G3 50,401 CCITT G4 1/ICCBased 54,329
02 黑白 TIFF 3315×2334 CCITT G4 35,518 CCITT G4 1/ICCBased 36,110
03 彩色 JPEG 格式 TIFF 512×384 DCTDecode 24,428 DCTDecode 8/ICCBased 21,753
04 灰度 JPG 445×600 DCTDecode 34,167 FlateDecode 8/Indexed 200,404
05 彩色 JPG 1024×768 DCTDecode 102,776 DCTDecode 8/ICCBased 71,540
06 16 级灰度 GIF 800×1199 LZWDecode 124,738 FlateDecode 4/Indexed 128,925
07 256 色 GIF 130×129 LZWDecode 8,408 FlateDecode 8/Indexed 6,990
08 黑白 PNG 32×32 FlateDecode 164 CCITT G4 1/ICCBased 41
09 2 色彩色 PNG 32×32 FlateDecode 112 FlateDecode 1/ICCBased 21
10 256 级灰度 PNG 600×905 FlateDecode 289,059 FlateDecode 8/Indexed 286,947
11 16 级灰度 PNG 720×1053 FlateDecode 74,322 FlateDecode 4/Indexed 74,943
12 24 位色 PNG 350×560 FlateDecode 72,107 FlateDecode 8/ICCBased 79,351
13 15 位色 BMP 260×235 未压缩 122,266 DCTDecode 8/ICCBased 5,783
14 16 色 BMP 940×20 RLE 8,134 FlateDecode 4/Indexed 2,868
图像文件总长度（字节） 946,600
PDF 文件总长度（字节） 989,431

注 1. 图像 01.tif 经 ACDSEE 转成 PDF 后，图像物理表示的高度变成 2206，原因后面会加以解释。

注 2. 对于索引色（Indexed）图像，“数据流长度”仅包含图像数据流长度，不包含索引（调色板）数据流长度。严格说来这会造成一些误差，但影响不大。以下各表与此相同，不再特殊说明。

从表 1.1 可以看出，对于 ACDSEE 发送过来的数据流，Acrobat PDF 虚拟打印机进行如下处理：

黑白图像重新压缩为 CCITT G4 数据流。

灰度、索引色（调色板）图像压缩为 Flate（ZIP）数据流，色深（BitsPerComponent）不变。

非索引色（如 15 位色、24 位色）图像压缩为 DCT（JPG）或 Flate 数据流。似乎 Acrobat PDF 虚拟打印机能自动识别压缩为哪种数据流更有利，但压缩成 JPG 数据流时似乎质量系数很低：文件更小，质量更差。

考虑跨平台特性，所有色彩均表示为 ICCBased，并给出对照表。

这种处理带来的问题是：

对于有损压缩的灰度 JPG，转换后就成了无损压缩的数据流，必然导致文件长度的膨胀。

对于原来无损压缩的彩色图像（PNG、BMP），转换后可能成为有损压缩的 JPG 数据流，造成图像质量下降（从转换后的数据流长度看，重新压缩的 JPG 质量系数不会太高）。对于原来就是有损压缩的彩色 JPG 图像来说，由于是解码后再压缩，图像质量将会逐次衰减。

为了确认是否所有软件在使用虚拟打印机转换 PDF 时，均会对 JPG 图像进行再压缩，我进行了第二个试验：在 Word 2003 中插入用例 1 中的 13 张图像（Word 2003 不支持 03.tif），每张图像前面再插入一点文字（图像编号），然后打印到 Acrobat PDF 虚拟打印。限于篇幅，这里不列举具体结果（用例 1 是公开的，Word 2003、Acrobat 也不难找，想要较真的人可以自己动手试一下），仅说明结果：

对 01、02 两个 TIFF 文件，Word 转换成 CCITT G4 压缩，而且 01.tif 物理表示的高度未变，这 比 ACDSEE 强。但是这两个图像均被分解成了多个对象（01.tif 分成 2 个，02.tif 分成 8 个），相当于将原始图像切割成多个水平横条，每个对象代表一条。

对 04、05 两个 JPG 图像，Word 将原始文件完整地嵌入了 PDF 文件，但在结尾添加了一个\r（0AH）字符。显然，Word 并没有对原始 JPG 文件进行解码、再压缩。

对 06、07 两个 GIF 文件，Word 打印后均成为 ZIP 数据流，与 ACDSEE 相似。

对 08、09 两个 PNG 文件，Word 打印后成了 4 位索引色图像（BitsPerComponent＝4，Indexed），压缩算法仍然为 ZIP。10～12 三个 PNG 文件转换结果与 ACDSEE 相似。

13、14 两个 BMP 文件转换结果与 ACDSEE 相似。

从转换结果的对比来看，Word 和 ACDSEE 的打印结果在 TIFF、JPG、PNG 方面存在较大差距：

对于 TIFF 图像，Word 对图像进行了切割，这个估计与 Word 对图像的显示方式有关；另外 Word 没有改变图像的物理表示， 这与下面要谈的“直接将图像嵌入 PDF 文件”的转换软件类似。

对于 JPG 图像，Word 没有再压缩的过程，因此不会造成图像质量的衰减。

对于 PNG 图像，Word 似乎有自己的表示方法，存在解码、重新压缩的过程，并且将 BitsPerComponent 小于 4 的 PNG 转换成 BitsPerComponent 等于 4 的图像，这点与 CxImage 很象。

为了搞清为什么 Word 不需要对 JPG 图像进行解压即可直接打印到虚拟打印机，我查了一下微软的 MSDN，其中对 StretchDIBits 函数的解释引起了我的注意：

Windows 98/Windows 2000: This function allows a JPEG or PNG image to be passed as the source image. How each parameter is used remains the same, except :

If the biCompression member of BITMAPINFOHEADER is BI_JPEG or BI_PNG, lpBits points to a buffer containing a JPEG or PNG image, respectively. The BITMAPINFOHEADER's biSizeImage member specifies the size of the buffer. The iUsage parameter must be set to DIB_RGB_COLORS. The dwRop parameter must be set to SRCCOPY.

To ensure proper metafile spooling while printing, applications must call the CHECKJPEGFORMAT or CHECKPNGFORMAT escape to verify that the printer recognizes the JPEG or PNG image, respectively, before calling StretchDIBits.

StretchDIBits 是在绘制、打印图像时的常用函数，虽然我不可能看到 Word 的源代码，但我相信 Word 直接或间接使用了这个函数。为了证实 Acrobat 虚拟打印机对 JPG 数据流的支持，我用 VC++写了一个小程序，核心打印代码来自 MSDN 文章“Testing a Printer for JPEG or PNG Support”，最终证实了我的猜测：Acrobat 虚拟打印机允许直接将 JPG 数据流发送给它，但是不支持 PNG 数据流。

综上所述，对于基于虚拟打印原理实现的图像转 PDF 工具，可能会存在如下问题：

对于有损压缩的 JPG 文件，转换成 PDF 后的质量与发出打印命令的软件密切相关。如象 ACDSEE 这样先解码再打印，必然会因为图像的再压缩而造成质量衰减或文件膨胀。而象 Word 这样直接将 JPG 数据流发送到虚拟打印机，则与软件内部的打印设置有关，设置好了可以直接将数据流完整嵌入 PDF 而不造成损失或膨胀，设置不好则同样可能造成损失。另外打印机对 JPG 数据流的支持受平台限制，我猜这就是为什么包括 ACDSEE 在内的大多数软件宁愿先解码后打印的原因：解码成 bitmap 再打印可以不受平台限制。

对于无损压缩的图像文件，如 GIF、PNG、BMP 等，真彩图像往往会被转换成有损压缩的 JPG 数据流，造成图像质量损失；灰度、索引色图像往往会被解码后再压缩成某种无损压缩数据流，如果虚拟打印机所选压缩算法的压缩效低于原图像压缩算法，则可能造成 PDF 文件的膨胀。

直接将图像嵌入 PDF 的转换软件工作原理与基于虚拟打印机的转换软件不同，其工作过程为：

用户在转换软件中选择需要转换的图像文件。

转换工具按照 PDF 文件规范创建 PDF 文件，写入文件头信息。

转换工具逐一从图像文件中抽取图像数据，视需要对数据进行转换，然后将数据打包成 PDF 对象，写入 PDF 文件。

转换工具写入 PDF 文件尾，打包结束。

为了测试图像嵌入式转换工具的效果，我将与表 1.1 相同的图像文件（用例 1），用 ACDSEE 8.0 (Build 39)的 Create PDF Wizzard 转换成 PDF，结果见表 1.2。

表 1.2 ACDSEE 8 PDF 创建插件转换结果

原始图像 PDF 中的图像数据

序号 说明 宽×长

（象素） 解码器 文件长度

（字节） 解码器 BitsPerComponent

/ColorSpace 数据流长度

（字节）

01 黑白 TIFF 1728×1103 CCITT G3 50,401 DCTDecode 8/DeviceGray 905,917

02 黑白 TIFF 3315×2334 CCITT G4 35,518 DCTDecode 8/DeviceGray 693,129

03 彩色 JPEG 格式 TIFF 512×384 DCTDecode 24,428 DCTDecode 8/DeviceRGB 34,496

04 灰度 JPG 445×600 DCTDecode 34,167 DCTDecode 8/DeviceGray 59,338

05 彩色 JPG 1024×768 DCTDecode 102,776 DCTDecode 8/DeviceRGB 129,655

06 16 级灰度 GIF 800×1199 LZWDecode 124,738 DCTDecode 8/DeviceRGB 319,743

07 256 色 GIF 130×129 LZWDecode 8,408 DCTDecode 8/DeviceRGB 6,706

08 黑白 PNG 32×32 FlateDecode 164 DCTDecode 8/DeviceGray 936

09 2 色彩色 PNG 32×32 FlateDecode 112 DCTDecode 8/DeviceRGB 896

10 256 级灰度 PNG 600×905 FlateDecode 289,059 DCTDecode 8/DeviceGray 185,845

11 16 级灰度 PNG 720×1053 FlateDecode 74,322 DCTDecode 8/DeviceGray 206,121

12 24 位色 PNG 350×560 FlateDecode 72,107 DCTDecode 8/DeviceRGB 38,468

13 15 位色 BMP 260×235 未压缩 122,266 DCTDecode 8/DeviceRGB 8,209

14 16 色 BMP 940×20 RLE 8,134 DCTDecode 8/DeviceRGB 22,018

图像文件总长度（字节） 946,600 PDF 文件总长度（字节） 2,619,592

从表 1.2 中可以看出，ACDSEE 8.0 (Build 39)的 Create PDF Wizzard 对图像的转换原则是：

灰度图像一律转换成灰度 JPG 数据流。

彩色图像一律转换成彩色 JPG 数据流。

看来 ACSEE 对它的 JPG 转换引擎还真不是一般地有信心！但从表 1.2 所列数据看，这种转换也不是没有问题：

对于黑白图像，CCITT G4 的压缩比通常比灰度 JPG 高许多，毕竟它是专为压缩黑白图像而研发的压缩算法。因此将 CCITT G3/G4 转换成灰度 JPG 无疑将会造成文件膨胀，而且膨胀得很明显。这点对电子文档来说比较重要：大多数白纸黑字的纸质文档扫描后都是黑白图像。

对于低色深（如 16 级灰度）图像来说，转换成灰度 JPG（256 级灰度）同样可能造成文件膨胀。

对于本来就是 JPG 压缩格式的图像，用 ACSEE 转换后也会出现文件膨胀的问题，莫非是 ACSEE 转换插件用的 JPG 质量系数比较高？

不论原来的图像格式是什么，经过 ACSEE 的转换插件转换后全部解码再重新压缩成有损的 JPG 数据流，无疑会对图像质量造成损伤。

ACSEE 的转换插件效果很令我失望，为了比较其它嵌入式工具的转换效果，我又用用例 1 测试了 verypdf 的 Image2Pdf v1.7，转换结果见表 1.3。

表 1.3 Image2Pdf 转换结果

原始图像 PDF 中的图像数据

序号 说明 宽×长

（象素） 解码器 文件长度

（字节） 解码器 BitsPerComponent

/ColorSpace 数据流长度

（字节）

01 黑白 TIFF 1728×1103 CCITT G3 50,401 CCITT G4 1/DeviceGray 41,638

02 黑白 TIFF 3315×2334 CCITT G4 35,518 CCITT G4 1/DeviceGray 34,981

03 彩色 JPEG 格式 TIFF 512×384 DCTDecode 24,428 DCTDecode 8/DeviceRGB 32,815

04 灰度 JPG 445×600 DCTDecode 34,167 DCTDecode 8/DeviceGray 34,167

05 彩色 JPG 1024×768 DCTDecode 102,776 DCTDecode 8/DeviceRGB 102,776

06 16 级灰度 GIF 800×1199 LZWDecode 124,738 RunLengthDecode 4/Indexed/DeviceRGB 206,880

07 256 色 GIF 130×129 LZWDecode 8,408 RunLengthDecode 8/Indexed/DeviceRGB 13,380

08 黑白 PNG 32×32 FlateDecode 164 FlateDecode/PNG 1/DeviceGray 91

09 2 色彩色 PNG 32×32 FlateDecode 112 FlateDecode/PNG 1/Indexed/DeviceRGB 21

10 256 级灰度 PNG 600×905 FlateDecode 289,059 FlateDecode/PNG 8/DeviceGray 288,582

11 16 级灰度 PNG 720×1053 FlateDecode 74,322 FlateDecode/PNG 4/Indexed/DeviceRGB 74,063

12 24 位色 PNG 350×560 FlateDecode 72,107 FlateDecode/PNG 8/DeviceRGB 71,954

13 15 位色 BMP 260×235 未压缩 122,266 DCTDecode 8/DeviceRGB 8,707

14 16 色 BMP 940×20 RLE 8,134 DCTDecode 8/DeviceRGB 20,890

图像文件总长度（字节） 946,600 PDF 文件总长度（字节） 942,458

从表 1.3 看，Image2Pdf 对图像数据的处理要比 ACDSEE 的 PDF 创建插件智能得多：

对于黑白 TIFF 文件，能够自动压缩成 CCITT G4；彩色 TIFF 解码后压缩成 JPG。

对于 JPG 文件，根本就没有解压、再压缩的过程，直接将原始 JPG 文件一个字节不改就嵌入 PDF 文件，从而避免因为再次压缩而造成质量衰减，而且解压、再压缩的时间也省了。

对于 GIF 文件，解压后压缩为 RLE（行程编码）。由于 RLE 的压缩率远不如 GIF 本身的 LZW 算法，因此这种再压缩会造成文件膨胀。估计这种吃力不讨好的做法与传说中 LZW 压缩算法的版权有关。

对于 PNG 文件，数据流压缩算法不变（PNG 压缩算法在 PDF 中对应 /DecodeParms[<</Predictor 15>>]），但数据流长度会稍小一些，估计是去掉了 PNG 文件中的无关信息。

对于彩色 BMP 文件，全部重新压缩成 JPG 数据流，在色彩数较多、色调过渡自然时能够减小文件长度，否则会增加文件长度。当然不论哪种情况画面质量都会损失。

其它图像转 PDF 的软件我还试过一些，不过基本与以上几种工具类似，都可能因为对图像数据流重新压缩而产生一些问题，差别只在问题的多与少、严重

与不严重：

将无损压缩转换成有损压缩，或对有损压缩解码后再次有损压缩，必然造成图像质量下降。

改变文件数据流的压缩方法，在某些情况下可以减小文件长度，在某些情况下则会造成文件长度膨胀。关键是看数据与压缩方法的搭配是否合适。

对于直接读取图像数据的转换工具，由于可以从原始图像文件中获取丰富的图像信息，包括原始数据压缩算法等，因此可以针对不同的文件格式或不同的图像情况做出选择；基于虚拟打印原理实现的转换工具，如果打印机只能得到解码后的数据流，选择的余地就会小一些：只能从 bitmap 数据流中获取色深等信息，然后自行选择算法重新压缩数据。

2、阅读的顺畅性问题

这里说的阅读顺畅性问题，是指：

如果 PDF 纸张选择 A4、B5 等标准纸张，而原始图像的长宽比例与所选纸张的长宽比例不一致，必然会在上下或左右出现较多空白，影响阅读。

如果 PDF 纸张随图像大小而变化，则转换出来的页面可能大小不一，在阅读时感觉页面跳来跳去，很是不爽。

对于第一个问题，目前没有什么好的解决方案。对于第二个问题，可能的解决方案包括：

建议用户在阅读时将 PDF Reader 设置为单页、适合宽度，这样每一页都会自动缩放到 Reader 的窗口宽度。如果嫌麻烦，也可以在生成 PDF 时就指定“初始视图”中的“页面布局”为“单页”,“放大率”为“适合宽度”。这种方法的缺点是前后翻页时不如“连续”模式顺畅。

在生成 PDF 文件时为页面指定一个固定宽度，页面的长度按照原始图像的长宽比自动伸缩。这种方法能保证在以“连续”模式阅读时页面不会跳来跳去，当然打印出来还是会在纸张的上下或左右产生空白。

3、对特殊图像格式的支持问题

这里说的“特殊图像格式”，其实主要就是 TIFF 格式。在常见的图像格式中，JPG、GIF、PNG、BMP 等都有严格的格式规定，可以发挥的余地不多。但是对于 TIFF 来说，由于标准本身希望能够包容尽可能多的东西，但是对实现细节没有给出具体的规定，所以各家软件生成的 TIFF 文件五花八门，令人头疼。

以我提供的测试用例 2 为例，这个其实是支持 TIFF 文件最权威的开源项目 libtiff 3.7.1 版所带测试图片，不过去掉了一张 caspian.tif（该图片共 3 通道，单通道采样位数高达 64 位浮点数，我的 32 位真彩显示器单通道采样位数只有 8 位整数，显示不了这么高级的图片）。但仅凭剩下的这些图片，已经可以难倒包括 verypdf 的 Image2Pdf 在内的一大批图像转 PDF 软件，就算是 ACDSEE 这样“专业”的图像浏览器，5.0.1 版在看这些图像时也会出现比例失调（fax2d.tif、g3test.tif）、看不了（quad-tile.tif）、颜色失真（smallliz.tif、zackthecat.tif）等问题；8.0 版虽然修正了上述问题，但又出现新的问题：看 dscf0013.tif 时颜色失真。

其实这些文件还算好，毕竟是 libtiff 组织提供的，至少它自己的源代码还能解出来。但在我接触到的国内专业扫描外包公司中，大多数公司提供的 TIFF 文件只要采用了有损压缩，多半就连 libtiff 也解不开，ACDSEE 更是想都不用想。有些甚至连专门显示 TIFF 文件的 Microsoft Office Document Imaging（微软 Office 2003 所带附件之一）都解不开。

偏偏由于工作需要，我必须和这些怪异 TIFF 文件打交道。我想到的出路包括：

不要在 TIFF 文件中使用有损压缩，尤其不要用各品牌专业高速扫描仪所带扫描软件生成有损压缩 TIFF 文件。由于历史原因，这些软件遵循的多半是古老的 TIFF 标准，生成的文件大概只有它们自己的阅读软件能打开。如果有必要对图像进行有损压缩，直接存储为标准 JPG 格式即可，这个很难玩什么花样。

以 libtiff 源代码为基础，针对这些图像不规范的地方，逐步修正 libtiff 代码。这就是为什么前段时间我自己写的 ComicEnhancer Pro 一直在升级的原因。我甚至怀疑，可能就是因为 TIFF 格式支持起来太麻烦，所以 IE 才不支持。

除 TIFF 外，PNG 文件也是一种可能会造成潜在麻烦的格式。但是与 TIFF 不同，PNG 的麻烦不在于文件格式本身或数据压缩算法，而在于它丰富的色彩表示：PNG 支持灰度、索引、彩色、Alpha 通道彩色图像，并且色深除低端的 1、2、4、8 位外，还支持 16 位色深。有兴趣又喜欢较真的人，可以到 libpng 下载一份 libpng 源代码，里面的 contrib\pngsuite 文件夹下就包含了一堆图片，专门用于测试软件对 PNG 色彩支持的能力。

从我测试的结果看，软件在处理 PNG 图像时可能出现的问题包括：

将 16 位色深简化成 8 位色深。这个在通常 24 位/32 位显示器上看不出问题，因为这些显示器最多只支持 8 位色深，但是将来高端显示成本降低后可能就会被人看出差异了。PDF 文件也是从 PDF 1.5 版（Acrobat 6.0）开始才支持 16 位色深。

对于低色深（BitsPerComponent < 4）的 PNG 图像，转换成 BitsPerComponent ＝4 的索引色图像，造成轻微的文件膨胀。

综上所述，目前图像转 PDF 工具普遍存在一些共性的问题，包括图像数据流重新压缩造成的问题、生成的 PDF 文件的阅读顺畅性问题、对特殊图像格式的支持问题等。为了更好地理解这些问题，并找到解决办法，下面先简单介绍一下 PDF 中与图像相关的基本概念。

二、预备知识

事先声明：本部分所有内容均来自 Adobe 公司发布的《PDF Reference 5th edition》， 说白了就是我看这份文档时做的笔记的一部分，所以看起来可能有点无头无尾，各位看得懂就看，看不懂就去看《PDF Reference 5th edition》原文吧。

1、PDF 支持的图像格式

在 PDF 文件中，图像点阵信息以压缩数据流的形式存在，PDF 通过过滤器（filter）对数据流解码。在《PDF Reference 5th edition》中，共介绍了十种过滤器，其中与图像相关的如表 2.1 所示。

表 2.1 PDF 文件支持的图像过滤器

过滤器名称	对应压缩算法通称	对应图像格式	压缩类型	说明
LZWDecode	LZW	GIF、TIFF	无损	通常用于索引色（调色板）图像
FlateDecode	ZIP	PNG、TIFF	无损	除图像外，也用于文本压缩
RunLengthDecode	RLE	BMP、TIFF	无损	通常用于单色图像
CCITTFaxDecode	G3/G4	TIFF	无损	专为黑白图像研发的高效压缩算法
JBIG2Decode	JBIG2	JBG	无损	专为黑白图像研发的高效压缩算法
DCTDecode	JPEG	JPG、TIFF	有损	用于 256 级灰度、24 位真彩自然图像
JPXDecode	JPEG2000	J2K、JP2	有损/无损	JPEG 的最新标准，压缩比与质量并重

从表 2.1 看，其实对大多数常见图像格式，都可以将原数据流直接嵌入 PDF 文件，不需要再重新编码。当然某些数据，如 JPG 文件中的注释、PNG 文件的文件头/文件尾，在 PDF 文件中没用，可以先剔除再将剩余部分嵌入 PDF 文件。而对于 TIFF 文件，需要针对具体压缩算法，将真正图像数据抽取出来再嵌入 PDF 文件。

直接使用原始数据流而不再重新编码，不仅能够节省图像转换成 PDF 的时间，而且对于有损压缩，可以避免因为反复压缩而造成图像质量的衰减。但是对于 GIF 格式的 LZW 压缩来说，情况有点复杂：由于 Unisys 公司声称对 LZW 算法拥有专利权，导致很多软件，包括大名鼎鼎的 xpdf 在内，放弃 对 LZW 的支持，改用开源的 Flate 压缩算法。Flate 其实是 Winzip 等软件使用的 ZIP 压缩算法的另外一个名字。《PDF Reference 5th edition》中对这两种算法的描述如下（黑体效果是我自己加的）：

Because of its cascaded adaptive Huffman coding, Flate-encoded output is usually much more compact than LZWencoded output for the same input. Flate and LZW decoding speeds are comparable, but Flate encoding is considerably slower than LZW encoding.

由于上文中黑体部分的描述，及诸多公司、组织卷入与 Unisys 的专利纠纷，因此在我见到的图像转 PDF 工具中，没有使用 LZW 压缩算法的，都宁愿将用 LZW 压缩的 GIF 图像解码后再压缩成 Flate 数据流。这种转换在多数情况下能获得更好的压缩比，但例外总是有的（所以上文用的词是 usually，而不是 always），如用例 1 中的 06.gif，LZW 就比 ZIP 有效得多，这样的图片我还有几张，不过限于篇幅和空间，就不节外生枝了。

对于 LZW 和 Flate 压缩，PDF 还支持预报器（Predictor），预报器表示根据

图像的某些特征，先对图像进行某些预处理后再对处理结果进行压缩，以获取更高的压缩比。在《PDF Reference 5th edition》中定义的预报器见表 2.2。小于 10 的预报器称 TIFF 预报器，源自 libtiff；10 以上的称 PNG 预报器，源自 libpng。因此如果 PDF 文件中定义图像的物理表示时使用了 Predictor 属性，多半可以猜到原始图像的格式。 在表 3.1 和表 1.3 中，为了更好地进行比较，采用 PNG 预报器的 Flate 解码器均标注为 FlateDecode/PNG。

表 2.2 预报器

预报器值	含义
1	No prediction (the default value)
2	TIFF Predictor 2
10	PNG prediction (on encoding, PNG None on all rows)
11	PNG prediction (on encoding, PNG Sub on all rows)
12	PNG prediction (on encoding, PNG Up on all rows)
13	PNG prediction (on encoding, PNG Average on all rows)
14	PNG prediction (on encoding, PNG Paeth on all rows)
15	PNG prediction (on encoding, PNG optimum)

2、图像在 PDF 中的物理表示

一幅图像在 PDF 文件中通常用一个 XObject 对象表示（某些 TIFF 图像可能要用多个对象表示），这个对象描述图像的原始象素点阵信息，因为这些点阵信息由产生图像的设备本身的物理性质（如扫描仪的 DPI、 数码相机的有效象素数等）决定，因此在这里称为图像的物理表示，在《PDF Reference 5th edition》中又称为采样表示（Sample Representation）。

要描述图像的物理表示，需要提供下列信息：

图像的宽度（width），以象素为单位。
图像的高度（height），以象素为单位。
每象素的颜色通道数，或色彩空间（The number of color components per sample, ColorSpace）。
每通道的采样位数（The number of bits per color component, BitsPerComponent）。
图像象素点阵数据流（stream）。
解码图像数据流所需的过滤器（Filter）名称，及过滤器采用的预报器（Predictor）。
以用例 1 为例，在 ACDSEE 8 PDF 创建插件转换的 PDF 中，用如表 2.3 所示的 XObject 定义第二 幅图像（数据来自 PdfView，右侧双斜杠后面是我自己加的注释）：

表 2.3 图像对象定义实例
9 0 obj

```
<<
/Type /XObject
/Subtype /Image
/Name /Image90
/Width 3315
/Height 2334
/BitsPerComponent 8
/ColorSpace /DeviceGray
/Filter [/DCTDecode]
/Length 693129
>>
stream
……
endstream
endobj // 对象定义开始，对象 ID 为 9
// 字典（dictionary）定义开始
// 对象类型为 XObject
// 对象子类型为图像
// 对象名称 Image90
// 图像宽度 3315 象素
// 图像高度 2334 象素
// 每通道采样位数为 8
// 色彩空间为 256 级灰度
// 解码过滤器为 JPG（参见表 2.1）
// 数据流长度 693129 字节
// 字典定义结束
// 数据流开始
// 数据流内容，一串 16 进制数，此处从略
// 数据流结束
// 对象定义结束
```

PDF 中的每个对象均有一个 ID（编号），通过对象 ID，可以对对象本身进行引用。如一个 LOGO 图像可能作为背景出现在每一个页面上，在每一页中没有必要都包含这个 LOGO 图像的实际数据，只要引用这个 LOGO 图像的 ID 即可。这样无疑可以提高 PDF 文件的存储效率。上例中图像的对象 ID 就是 9。

3、图像在 PDF 中的逻辑表示

前面说的图像的物理表示是用象素点来表示图像，但是如果直接按照象素点对图像进行显示、打印，可能会出现问题。以用例 1 中的第二 幅图像为例，象素点阵为 3315×2334，如果在分辨率为 96 DPI 的显示器上显示，尺寸是 34.5 英寸×24.3 英寸（1 英寸＝2.54 厘米，实际英寸数＝象素数÷DPI，如 3315÷96=34.5 英寸），而在分辨率为 300 DPI 的打印机上打印，打出来只有 11.1 英寸×7.8 英寸，

这显然与 PDF 要求的“在任何平台上均可获得相同的效果”不符。因此在用物理表示定义出图像的象素点阵后，在实际需要显示图像的地方，不仅要给出图像物理表示的对象 ID，还需要给出图像的逻辑表示，包括：

图像的逻辑尺寸。这个尺寸的单位是 1/72 英寸，因此是一个逻辑概念，即不论在什么样的设备上输出图像，图像的大小都是固定的英寸值，而不会随着输出设备的 DPI 值而变化。

图像的偏移量，即图像左上角点距离页面左上角点的距离，这同样是一个与设备的物理分辨率无关的逻辑量，单位为 1/72 英寸。

图像的旋转角度。

这种物理与逻辑表示的分离，可以带来一些好处：

同一份物理数据，可以在不同的地方、用不同的大小、以不同的旋转角度进行显示。

通过将物理表示映射成逻辑表示，可以脱离设备的物理性能限制，在不同的设备上获得相同的效果。

具体到 PDF 文件格式上，在一个页面上显示一幅图像，除了前面说过的图像的物理表示对象外，还需要定义页面（Page）对象，然后在 Page 对象中：

用 MediaBox 属性定义页面的逻辑大小，单位为 1/72 英寸。

用 Resources 属性定义页面中包含的资源，即前面说的图像物理表示的对象 ID。

用 Contents 属性定义资源对象（图像）的逻辑表示。

Contents 属性通常定义一个六元组，表示为[a, b, c, d, e, f]，则从图像物理坐标(x, y)映射为逻辑坐标(x', y')的映射关系可以表示为如下矩阵运算：

$$[x'\ y'\ 1] = [x\ y\ 1] \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 0 \end{bmatrix} \qquad \text{式 2.1}$$

或表示为如下解析表达式：

$$x' = ax + cy + e \qquad \text{式 2.2}$$
$$y' = bx + dy + f$$

从《计算机图形学》知识可知，式 2.1、式 2.2 中参数 a、d 分别为 x、y 向比例系数，实现从物理尺寸到逻辑尺寸的映射；c、b 为旋转系数，表示图像显示时的旋转角度；e、f 为平移系数，表示图像到页面左上角的偏移量。

以用例 1 为例，在 ACDSEE 8 PDF 创建插件转换的 PDF 中，用如表 2.4 所示的结构定义第二页。

表 2.4 页面对象定义实例

8 0 obj

```
<<
/Type /Page
/Parent 3 0 R
/Contents 7 0 R
/MediaBox [0 0 3315 2334]
/Resources <</ProcSet [/PDF /ImageC] /XObject <</Image90 9 0 R>>>>
>>
endobj // 对象定义开始
// 字典定义开始
// 对象类型
// 父对象
// 内容在对象 7 定义
// 页面大小
// 页面包含的图像
// 字典定义结束
// 对象定义结束
```

内容对象 7 中定义了图像对象的逻辑表示，如表 2.5 所示。

表 2.5 图像对象的逻辑表示实例

```
7 0 obj
<<
/Length 38
>>
stream
q
3315 0 0 2334 0 0 cm
/Image90 Do
Q
endstream
endobj // 对象定义开始
// 字典定义开始
// 数据流长度 38 字节
// 字典定义结束
// 数据流开始
// 保存图像状态(Save graphics state)
// 式 2.1 座标映射所需的六元组(Coordinate transformation Matrix)
// 绘制映射后的图像(Paint image)
// 恢复图像状态(Restore graphics state)
// 数据流结束
// 对象定义结束
```

从表 2.5 的六元组参数看，ACDSEE 8 PDF 创建插件用一种很偷懒的方法构造该六元组：直接用图像的物理象素尺寸作为逻辑尺寸。由于屏幕 DPI 通常为 96 DPI，而 PDF 为 72 DPI，这种偷懒造成的后果就是图像在 PDF Reader 中按照“实际大小”显示时，看起来会比在 ACDSEE 中按照“完整大小”显示更大一些。精确一点说，这张图片在 PDF 中的“实际大小”达到了 46.04 英寸×32.42 英寸。其中 46.04=3315÷72，32.42＝2334÷72。

对于喜欢较真的人来说，ACDSEE 的这种偷懒造成了更深层次的失真：用例 1 中的第二幅图像是一张扫描形成的 TIFF 图像，在 TIFF 文件结构中记载了扫描时扫描仪使用的 DPI 值——200 DPI。在 ACDSEE 8 中打开此图像文件，点击“文件->属性”菜单，在显示出来的文件属性中即可看到扫描 DPI，及按照扫描 DPI 换算，这张图片对应原始纸质页面的大小——16.57 英寸×11.67 英寸。这个尺寸与 46.04 英寸×32.42 英寸相比，实在是差得太远了一点。

而如果用 verypdf 的 Image2Pdf v1.7 对同一张图片进行转换，可以看出转换后的 PDF 页面大小为 16.57 英寸×11.67 英寸，即在 PDF Reader 中选择按照“实际大小”进行显示，显示出来的图像大小，正好与原始纸质文件的大小一模一样。显然这样的结果更符合文档电子化的要求和习惯。

Image2Pdf 的这种“保真”转换过程可以描述如下：

如果图像文件中记录了扫描时扫描仪的 DPI 设置，则先按照“英寸数＝象素数÷扫描 DPI”，计算出图像的原始尺寸，以英寸为单位。

按照“PDF 尺寸＝英寸数×72”，将以英寸为单位的图像原始尺寸转换成 PDF 中的逻辑尺寸，并据此填写六元组。

三、问题的解决办法

在了解了相关预备知识后，再回顾前面提到的图像转 PDF 需要面对的问题，其答案自然明了：

图像数据流重新压缩造成的问题：对有损压缩图像数据，应尽量将原始数据流嵌入 PDF 文件，避免重新压缩造成图像质量衰减；对无损压缩图像数据，可以根据图像特征选择合适的无损压缩算法重新压缩图像数据，以节省存储空间，也可以直接将原始图像数据嵌入 PDF，以节省重新压缩所需的时间。

阅读的顺畅性问题：提供灵活多样的页面布局供用户按需选择，包括固定纸张大小、固定纸张宽度、按照图像大小定制页面等。页面大小的不同不应对原始图像数据流（图像的物理表示）造成影响，而是通过定义图像的逻辑表示，由 PDF Reader 本身来完成必要的图像缩放工作。

特殊图像格式的支持问题：这个问题靠等待很难等到结果，最简单的办法就是自己去面对、解决。

总之，对于象我这样有特殊要求的人来说，在目前的情况下要想得到满意的结果，还是只能贯彻“人要靠自己”的原则。当然也没有必要重新发明轮子，在我看来最理想的情况就是能够在现有开源项目基础上，通过必要的修改和补充，就能达到我的要求。但是 google 的结果令我稍微有点惊讶：虽然目前最权威的

图像 codec 开源项目都是基于 C 的，包括 JPEG LIB、libpng、libtiff 等，但是偏偏在 PDF 生成领域，JAVA 似乎比 C 多，包括 iText 等一大批开源项目，而 C 只有 PDFlib、ClibPDF、Panda 等有数的几个。

ClibPDF 从参考手册上看，在绘图等功能方面很有特色，但是对于图像文件的支持较差，而且要付费，所以我粗看了一下，没有深究。

Panda 的功能、接口都非常简洁明了，生成的 PDF 文件也没多少费话，所以我花时间详细试了一下。结果发现一个小小的小缺点：它的内存漏洞实在是太多了点，补都补不过来，最后只好放弃。

相比之下，PDFlib 堪称内容丰富、功能强大，某些高级功能（web 优化、权限控制等）虽然要付费才能看到，但就算是免费开源出来的 PDFLib Lite，对于各种图像格式的支持也足够一般性使用了，而且基本上没什么明显的内存漏洞。因此我在 DACapturer 中试用了一把。但是在使用一段时间后就发现一些问题：

PDFLib 的强大其实是和它代码的复杂程度分不开的，想对这样的代码做更改，实在太麻烦了。

PDFLib 生成的 PDF 文件中废话太多，导致文件膨胀。DACapturer 对此不是很在乎，但是在需要处理的文件数以万、十万做单位的业务系统中就需要考虑了。

PDFLib 对 JPG、PNG 的支持很好，但是对 TIFF 的支持可以用“令人发指”来形容：不支持 JPEG/OJPEG 压缩的 TIFF 还可以弥补（俺在 DACapturer 中通过在 PDFLib Lite 之外打补丁的办法解决），但是将 TIFF 中的每个 strip 当作一个对象来处理，就有点过分了，甚至发生过这样的事：用 PDFLib 转换十几页 TIFF 成为 PDF，居然在 PDF 中创建了几千个 obj，后来用 iText 对这样的 PDF 文件进行合并操作， 甚至活活把 iText 拖垮了。所以用了没多久，俺就下定决心一定要和 PDFLib 说再见。

由于问题的根子出在 TIFF 文件上，所以我的目光很快集中到 llibtiff 源代码中包含的 tiff2pdf.c。这份代码是我见过最简洁的 PDF 生成代码，而且由于是 libtiff 自己提供的，马马虎虎算是出身名门、血统纯正，对 TIFF 文件的支持当然很可观，我就是用它弥补了 PDFLib 不支持 JPEG/OJPEG 压缩 TIFF 文件的问题。当然这份代码也不是一点问题没有：

为了节省内存消耗，生成 PDF 时采用了一种很少见的顺序，导致在生成过程中难以动态添加新的图像。

对 JPEG/OJPEG、CCITT G3/G4、zip 压缩的 TIFF 支持很好，但是对其它格式 TIFF 的支持有待加强，用用例 2 测试一下就知道了。

好在这份代码比较简单，结构中规中矩，改起来不是太难。最终我以它为基础实现了新版图像转 PDF 内核，并结合大名鼎鼎的 CxImage，将对图像格式的支持从 TIFF 扩展到了 JPG、PNG、BMP 和 GIF，成为公开发行的免费软件 FreePic2Pdf，能够实现：

对有损压缩的 jpg 文件及采用 JPEG/OJPEG 算法压缩的 TIFF 文件，能直接

将原始 JPEG 数据流嵌入 PDF 文件，避免因为重新采样而造成图像质量下降。

对于无损压缩的图像文件，黑白图像解码后压缩为 G4，其它解码后压缩成 ZIP 数据流嵌入 PDF 文件。虽然解码/压缩需要消耗一些时间，但是在多数情况下可以减小 PDF 文件长度。

可以指定生成的 PDF 文件的页面大小（除 A4、B5 等，还支持国内常用的 32 开、16 开、大 32 开）及页边距，这种指定不会更改图像的物理表示，只影响 PDF 中对图像的逻辑表示。

如果不指定页面的纸张大小，可以指定页面的固定宽度（长度随图像大小伸缩），保证连续阅读时不会因为页面宽度变来变去而影响阅读。

对 libtiff 源代码进行了更改，以尽可能支持各种特殊格式的 TIFF 文件；直接调用 libpng 对 PNG 图像进行处理，以支持特殊色深的 PNG 文件。

用 FreePic2Pdf 对用例 1 进行转换，结果见表 3.1。

表 3.1 FreePic2Pdf 转换结果

原始图像					PDF 中的图像数据		
序号	说明	宽×长（象素）	解码器	文件长度（字节）	解码器	BitsPerComponent /ColorSpace	数据流长度（字节）
01	黑白 TIFF	1728×1103	CCITT G3	50,401	CCITT G4	1/DeviceGray	41,638
02	黑白 TIFF	3315×2334	CCITT G4	35,518	CCITT G4	1/DeviceGray	34,981
03	彩色 JPEG 格式 TIFF	512×384	DCTDecode	24,428	DCTDecode	8/DeviceRGB	23,169
04	灰度 JPG	445×600	DCTDecode	34,167	DCTDecode	8/DeviceGray	34,167
05	彩色 JPG	1024×768	DCTDecode	102,776	DCTDecode	8/DeviceRGB	102,776
06	16 级灰度 GIF	800×1199	LZWDecode	124,738	FlateDecode	4/Indexed/DeviceRGB	126,407
07	256 色 GIF	130×129	LZWDecode	8,408	FlateDecode	8/Indexed/DeviceRGB	7,031
08	黑白 PNG	32×32	FlateDecode	164	CCITT G4	1/DeviceGray	40
09	2 色彩色 PNG	32×32	FlateDecode	112	FlateDecode	1/Indexed/DeviceRGB	17
10	256 级灰度 PNG	600×905	FlateDecode	289,059	FlateDecode	8/DeviceGray	278,021
11	16 级灰度 PNG	720×1053	FlateDecode	74,322	FlateDecode	4/Indexed/DeviceRGB	73,199
12	24 位色 PNG	350×560	FlateDecode	72,107	FlateDecode/PNG	8/DeviceRGB	71,954
13	15 位色 BMP	260×235	未压缩	122,266	FlateDecode/PNG	8/DeviceRGB	31,764
14	16 色 BMP	940×20	RLE	8,134	FlateDecode	4/Indexed/DeviceRGB	2,832

图像文件总长度（字节） 946,600 PDF 文件总长度（字节） 838,932

对比表 3.1 和表 1.3，可以看出 FreePic2Pdf 优先考虑图像质量，其次考虑压缩比、生成速度。

另外用例 1 的第一张图片很有趣，用 libtiff 带的 TiffInfo 查看它的信息如下：

TIFF Directory at offset 0xc3c6
Image Width: 1728 Image Length: 1103
Resolution: 204, 98 pixels/inch
Bits/Sample: 1
Compression Scheme: CCITT Group 3
Photometric Interpretation: min-is-white
FillOrder: lsb-to-msb
Orientation: row 0 top, col 0 lhs
Samples/Pixel: 1
Rows/Strip: (infinite)
Planar Configuration: single image plane
Page Number: 1-1
Software: fax2tiff
Group 3 Options: (0 = 0x0)
Fax Data: clean (0 = 0x0)
Bad Fax Lines: 0
Consecutive Bad Fax Lines: 0

这张图片在宽度方向上的扫描 DPI 约为长度方向的两倍（204/98），如果对这种差异处理不好，会带来意外的结果。以 ACDSEE 为例，5.0.1 版显示该图片时就会变形，页面顶部的圆变成了扁椭圆，到 8.0 版时显示出了正圆，但是图像的长度从 1103 变成了 2206，而且在 ACDSEE 8 打印和用 PDF 创建插件转换成 PDF 后，在 PDF 文件的图像物理表示中，这张图片的长度均描述为 2206 象素。显然，ACDSEE 内部对图像数据流进行了更改（沿长度方向放大一倍），以符合原长宽比，这对于图像显示软件来说无可厚非，但是对于 PDF 转换软件来说就有点多余，会增加最终 PDF 的文件长度。Image2Pdf 没有对这张图片的物理表示进行更改，而是试图通过调整图片的逻辑表示，由 PDF Reader 在显示时进行长宽比调整。但不幸的是，Image2Pdf v1.7 似乎把比例算反了，结果导致最终 PDF 显示出来后，圆变成了长椭圆。FreePic2Pdf 吸取了这些教训，能够通过对图像逻辑表示的正确设置，在不改变物理表示的情况下， 以正确的长宽比例显示该图像。

四、小结

由于种种原因，目前图像转 PDF 工具容易出现图像数据流重新压缩造成的问题、阅读的顺畅性问题、对特殊图像格式的支持问题等。

解决图像数据流重新压缩造成的问题的建议：对有损压缩的图像数据，应尽

量将原始数据流嵌入 PDF 文件，避免重新压缩造成图像质量衰减；对无损压缩图像数据，可以根据图像特征选择合适的无损压缩算法重新压缩图像数据，以节省存储空间 ，也可以直接将原始图像数据嵌入 PDF，以节省重新压缩所需的时间。

解决阅读的顺畅性问题的建议：制作工具提供灵活多样的页面布局供用户按需选择，包括固定纸张大小、固定纸张宽度、按图像大小调整纸张大小等。页面大小的不同不应对原始图像数据流（图像的物理表示）造成影响，而是通过定义图像的逻辑表示，由 PDF Reader 本身来完成必要的图像缩放。

对特殊图像格式的支持，需要针对具体情况进行开发。

为了验证我提出的上述问题及其解决方法，我开发了一个免费的图像转 PDF 工具 FreePic2Pdf，有需要的可以到我的网站下载。该软件考虑的优先顺序依次是：图像质量、PDF 文件大小、转换速度。

五、题外话一：PDF 转图像

前面说了半天图像转 PDF，自然会产生一个问题：将 PDF 转成图像又如何？

我个人认为目前将 PDF 转成图像也可以分成两种：

将 PDF 每一页的内容（包括图像和文字）转成一个图像文件，从感觉上类似于对 PDF Reader 的显示区进行截屏。

从 PDF 文件里找出原始图像数据流，然后转存成对应的图像文件。

第一种的代表软件包括 verypdf 公司的 PDF2HTML 等。PDF2HTML 除了将页面转成图像，还能生成包含图像和翻页按钮的 HTML 页面，方便在没有安装 PDF Reader 的机器上浏览原 PDF 文件的内容。不过在这种“眉毛胡子一把抓”的转换结果里要取出某幅图像的内容，大概只能用 Photoshop 慢慢抠了。

第二种的代表软件包括 Adobe Acrobat Professional（菜单项：Advanced->Export All Images）。如果喜欢开源代码，也可以看看 Xpdf 组织提供的 pdfimages。从我使用的结果看，Acrobat 略显霸道，不管原来的图像是什么格式，转换出来都成了一种格式。pdfimages 稍好一点，JPG 数据流 可以直接导出成 JPG 文件，其它无损数据流解码后导出为 ppm 文件，不过对于某些特殊色彩空间（ColorSpace）的 JPG 数据流，直接导出会导致偏色，只能解码后导出为 ppm 文件。 其实部分特殊色彩空间可以导出为 JPG 压缩的 TIFF 文件，从而避免对数据进行解码、再压缩，pdfimages 不知道为什么没有考虑。

六、题外话二：除了 PDF，还有什么？

以 IT 界的眼光来看，电子文档发展到现在历史也不算短了，而且由于巨大市场前景的诱惑，各厂家也都纷纷推出了自己的格式。单纯从以支持扫描图像为主的电子文档来说，格式虽多，但是能够成气候、形成标准的，除了 PDF 格式外，还有多页 TIFF、JBIG2、DjVu 等。这些格式的共同点是：

支持多页，能够将整卷档案或整部书存储在一个文件中。

遵循开放的标准，能够吸收最先进的图像压缩技术为己用。

当然这些格式目前的影响力都不如 PDF，我认为原因也都差不多：

宣传和市场工作做得不够。PDF 在成为 ISO 标准前有 Adobe 公司在花大力气推动，现在更有 N 家公司卷了进来，市场的大饼越做越大，相比之下其它格式就显得技术有余，市场不足。

相应的支持工具和软件不足。PDF 虚拟打印机用起来多方便，其它格式的虚拟打印机则少得多。就算是用专用工具辛辛苦苦做出来，想和其它人分享成果的时候，还得问问他的机器上有没有装相应的浏览软件，未免太麻烦。当然浏览的问题和前一个问题相关，几年前也不是所有机器上都装 PDF Reader 的。

1、多页 TIFF

这个应该算比较老的标准了，由于扫描、出版界传统上就习惯用 TIFF 格式，所以将多页 TIFF 作为电子文档的一种标准格式，应该是顺理成章的事，国内部分省市先行制定的电子档案管理相关规定也曾要求用多页 TIFF 作为扫描电子文件的存储格式。

但是从实际情况看，真正用多页 TIFF 存储的电子文档并不多，在 2005 年颁布执行的《中华人民共和国行业标准 DA/T31—2005 纸质档案数字化技术规范》中，干脆就没多页 TIFF 什么事：

8 图像存储

8·1 存储格式

8·1·1 采用黑白二值模式扫描的图像文件，一般采用 TIFF(G4)格式存储。采用灰度模式和彩色模式扫描的文件，一般采用 JPEG 格式存储。存储时的压缩率的选择，应以保证扫描的图像清晰可读的前提下，尽量减小存储容量为准则。

8·1·2 提供网络查询的扫描图像，也可存储为 CEB、PDF 或其他格式。

多页 TIFF 为何会遭到冷落呢？我猜测的原因包括：

缺乏方便的浏览工具。众所周知 IE 不支持 TIFF 格式，所以网上浏览 TIFF 只能借助专门开发的控件。即使只在本地机上浏览，也只有 ACDSEE 等为数不多的图像浏览软件支持多页 TIFF，浏览时想做标注、笔记很困难。

格式不规范，这个恐怕是最为致命的问题。从我接触的情况看，由于历史的、技术的和其它的原因，目前国内众多扫描外包服务公司提供的扫描 TIFF 文件，黑白图像用 G4 压缩不会有什么问题，但是灰度、彩色图像在有损压缩时多半都用 OJPEG 压缩，而且格式多与规范不符，这就造成扫描出来的图像只能用该公司提供的图像浏览软件才能浏览，极大地限制了 TIFF 文件的传播。俺在实际工作中为了处理客户遍布全国的分支机构委托当地外包商扫描的档案文件，与这些非标准 TIFF 文件进行了长期的、艰苦卓绝的斗争（看看俺的 ComicEnhancer Pro 最近的更新记录就知道了）， 相信有资格说这句话。我相信在《中华人民共和国行业标准 DA/T31—2005 纸质档案数字化技术规范》中规定灰度和彩色图像用 JPG 存储，也就是为了避免产生这些不规范的有损压缩 TIFF 文件。

但是对于 TIFF 格式的生命力，我个人从未表示怀疑：与某些静态图像格式

不同，TIFF 标准一直在与时俱进，不断将先进的图像压缩技术吸收进来，目前已经支持主流的 CCITT、JPEG、LZW、ZIP 等技术，新版本的草案中则计划包含对 JPEG 2000、JBIG2 等先进算法的支持。这些都让我充满期待。

2、JBIG2

这种格式专门针对以文字为主、黑白扫描的图像文件，属无损压缩，据称比 G4 压缩算法的压缩率高很多，目前已成为 ISO 标准，PDF 从 1.4 版（Acrobat 5.0）开始允许内嵌 JBIG2 图像，未来的 TIFF 标准也打算吸收 JBIG2 压缩算法。

JBIG2 的原理类似 OCR：先对图像进行分割、匹配，在识别出子图像（如文字）后，将整幅图像看作子图像及其位置的集合，存储时只存储子图像和子图像出现的位置，其它背景信息全部过滤掉，因此不仅能够提供很高的压缩比，而且能够实现类似文字检索的图像全文检索。

虽然前景诱人，但是我个人认为 JBIG2 目前还存在下列问题：

压缩率严重依赖于图像本身的内容和压缩引擎的模板表。对于字母文字来说，字母总数毕竟有限，因此重码率很高，自然压缩比也很高，但是对于中文来说，可能就没这么理想了。不过从我试用的情况看，至少不会比 CCITT G4 的压缩比差。

缺乏必要的代码支持，严重阻碍了该标准的推广普及。与其它图像格式不同，目前还没有一个开源组织提供真正的 JBIG2 压缩支持：Markus Kuhn 提供的 JBIG-KIT 只支持 JBIG1，并已经停止更新；jbig2dec 只提供解码代码（俺怀疑 PDF 的 JBIG2 解码代码就来自这里），不提供编码代码。

3、DjVu

这个也是针对扫描电子文档的，但是与 JBIG2 不同，针对的是彩色、图文混排的图像。

DjVu 的原理是先对图像进行分析，然后按照内容分层，包括背景层、文字层、图像层等，对不同的层使用不同的压缩算法和参数，以获得最好的图像质量和压缩比。

与 JBIG2 不同，DjVu 不仅有 djvuzone 组织在维护，而且有开源项目 DjVuLibre 作为支撑，因此现在不仅有不同平台下的编码、解码器，连查看 DjVu 文件的 IE 插件都发布了，未来应该大有希望。

4、双层 PDF

双层 PDF 是这样的 PDF 文件：PDF 文件的每一页都包含两层，下层是从纸质文件扫描出来的原始图像，上层是用 OCR 软件对扫描图像进行识别后产生的文字结果，但字体效果设置成透明。这样用户在阅读 PDF 文件时看到的是扫描图像，可以 100%保留原始版面效果（包括公章、签名），在需要的时候，又可以

通过 透明的文字信息支持选择、复制、检索等功能。

与普通 PDF 文件相比，双层 PDF 能够同时兼顾视觉效果和使用方便性，因此在国内办公、档案领域正在引起重视，我个人相信会有美好的“钱途”。

显然，双层 PDF 的内容检索、内容复制与 OCR 识别结果有直接的关系。先不说目前国内 OCR 软件的识别率如何，最关键的一点是目前没有任何一个中文 OCR 引擎是免费、开源的（英文的则有 gocr 等一批），所以双层 PDF 生成工具也都不是免费的，而是“面向企业市场”，我相信穷困的个人用户在不违法的情况下很难消受得起。

http://www.readfree.net/bbs/read.php?tid=292129&keyword=

文本PDG文件名构成

作者：马健

文本 PDG 的构成规则为：
<前缀><起始页号>_<页数>.pdg

前缀和图像版 PDG 的一样：
bok：书名页
fow：前言页
!：目录页
正文页没有前缀。

例如：
bok1_1.pdg：书名页，内含 1 页
fow1_6.pdg：前言页，内含 1～6 页
!1_3.pdg：目录页，内含 1～3 页
42_6.pdg：正文页，起始页 42，内含 6 页，下一个文件的起始页号为 42＋6＝48。通常情况下正文页每个文本 PDG 文件包含 6 页，但是也有例外，似乎与文件大小有关
144_8.pdg：正文页，起始页 144，内含 8 页，下一个文件的起始页号为 144＋8＝152
211_4.pdg：正文页，起始页 211，内含 4 页，下一个文件的起始页号为 211＋4＝215
希望有人能够据此写一个生成 InfoRule.dat 的软件，免得总有人问为什么用 Pdg2Pic 不能转换文本 PDG。

http://www.readfree.net/bbs/read.php?tid=4477700&keyword=

文本PDG转PDF

作者：马健

本帖被 nulc 设置为精华(2007-06-24)

声明：

1、谨以此文献给喜欢折腾的各位热血人士，不喜欢折腾的就不必看了。

2、既然喜欢折腾，就不要怕麻烦。“既当婊子又立牌坊”的好事也许有，但不一定会轮到你我头上。

3、本文欢迎转载，不过转载的时候请注明原作者为 strnghrs。

文本 PDG 的常规转换步骤：

1、下载，解密。

2、用 Pdg2Pic 解成散页 PDF。

3、用 Adobe Acrobat Professional 合并成一个 PDF。

由于文本 PDG 通常没有封面、版权等附属页，因此用上述步骤制作的 PDF，俗称“裸奔版”，与“折腾”的精髓实在相去甚远。为了给“裸奔版”穿上衣服，还需进行下列操作：

1、下载图像版封面、书名、版权等附属页，并解密。

2、往下载到的文件夹里扔一个名为 000001.pdg 的文件，别管文件内容是什么，只要是一个没有加密的图像版 PDG 文件即可，最好是 JPG（这样在步骤 7 中比较好定位），绝对不能用 T3 类多层 PDG。

3、用 Pdg2Pic 将附属页转换成图像，并生成 FreePic2Pdf.itf、FreePic2Pdf.txt、FreePic2Pdf_bkmk.txt。

4、打开前面用文本 PDG 转换、合并出来的 PDF，点“文件->属性”，查看页面宽度，然后按照下列公式折算成 FreePic2Pdf 中的宽度：

FreePic2Pdf 中的宽度=宽度（厘米数）÷2.54×96

例如 Acrobat 中显示的页面宽度为 13.510 厘米，则计算出来的宽度即为 511。如果在 Acrobat 里看到的宽度单位是英寸，则上面公式中的÷2.54 可以省略，成为：

FreePic2Pdf 中的宽度=宽度（英寸数）×96

5、打开 FreePic2Pdf.itf，将[Main]段 MinWidth 的值改成上面计算出来的宽度值，这样转换出来的图像 PDF 与原文本 PDF 的页宽相同。

6、将 FreePic2Pdf.itf 中的段名[TextPage]、[Bkmk]分别改成[TextPage1]、[Bkmk1]，这样生成图像版 PDF 时就不会把书签、说明等带进去。

7、删除前面扔进去的 000001.pdg 转换出来的图像文件，这通常是最后一个文件，除非有附录页。

8、用 FreePic2Pdf 将转换出来的图像文件合并成一个 PDF，没有书签、文本。

9、用 Adobe Acrobat Professional 将图像、文本版 PDF 合并成一个。

10、把步骤 6 中改变的段名再改回来。

11、用 FreePic2Pdf 挂书签、改页码、加说明。大功告成。

文本文件合并批处理(可以将 1,2,3...9 补 0 后，按子目录合并）

文本文件合并批处理

由于ｌｐ想将下载的 txt 小说合并后 copy 到手机中看，本来想用老马的 textforever 进行合并的，但是

textforever 不能处理文件名长度不一致的数字文件名，而且不能对当前目录中的各个子目录批量合并，于是写

了下面的批处理文件。

1. 先处理文件类似下面的文件名：

1.txt 10.txt 11.txt 12.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt 9.txt

这些文件合并后章节的顺序不对，需要将 1,2,3,..9 的文件名前补一个 0，将文件名排序成下面的样子：

01.txt 02.txt 03.txt 04.txt 05.txt 06.txt 07.txt 08.txt 09.txt 10.txt 11.txt 12.txt

2. 然后分别将当前目录下的每个子目录中 txt 文件合并到当前目录中，并且以子目录名命名文件名

具体用法：将下面的文字复制到一个文本文件中，再将文件保存成.bat 文件，然后将.bat 文件复制到需要处理的目录运行。

@echo off

cls

echo "更改文件名，将１～９的文件名前补０"

```
echo "注意：此项操作只能处理 1-9 的文件名"

pause

for /r %%i in (.) do for /l %%j in (1,1,9) do ren "%%~dpni\%%j.txt" 0%%j.txt

echo "将 1 ～ 9 的文件名前补 0 完成"

echo "文件合并开始"

pause

for /r %%i in (.) do copy "%%~dpni\*.txt" "%%~dpi%%~ni.txt"

echo "文件合并结束"

pause
```

http://www.readfree.net/bbs/read.php?tid=275822&keyword=

给清晰版PDG无损减肥

作者：马健

清晰版虽好，不过收集多了也占地方，所以我认为讨论一下怎么给它减减肥还是很有必要的。另外收集清晰版的目的本来就是为了质量，所以一切有损的方法都不在本文讨论之列，包括缩小图像尺寸等，如果您喜欢这样的方法，建议直接去下载快速版还更方便。

讨论之前，首先要从理论上解决一个问题：清晰版能不能再被无损压缩？

我的回答是：当然能，而且压缩空间还不小，只不过技术有点复杂。

我的理由如下：

对于清晰版来说，T1、T2、T3 的存储格式分别为

T1：CCITT

T2：JPG

T3：CCITT G4 ＋ JPG

首先说 CCITT G4，这个只要把它压缩成 DjVu，至少可以砍掉 20%的文件长度，而且还是无损，如果对于字母文字页采用有损 DjVu 还能压掉更多。

其次说 JPG。T3 里的 JPG 插图实在没有什么办法可想，但是其实大多数尺寸令人咬牙切齿的清晰版，都是 T2 格式的单层 JPG。对 T2 JPG 的减肥办法，就是把它分解成 T3，文字部分用 DjVu，插图无损切割到最小尺寸，还是用 JPG。

所以从理论上说，对清晰版无损减肥是可以做到的，不过有几个技术问题需要解决：

1、将 CCITT G4 转换成 DjVu，并封装成 PDG。这个好办，转换代码是现成的，PDG 文件 00H 格式也没有悬念，直接把 DjVu 数据流写在文件头后面就可以了。

2、将 T2 转换成 T3。这个最难，难就难在怎样将插图与文字切割开。对于

搞 OCR 的人来说，这个是必须过的第一关，其它人可能就会过不去，至少我现在就不知道怎么过。

3、在将插图识别出来后，将插图从整页 JPG 中无损切割下来。这个也好办，网上有开源的，网站为 http://jpegclub.org。

如前所述，减肥的理论和方法都已经具备了，缺的就是将插图与文字切割开的方法和代码。由于种种原因，我不能去钻研这种技术，如果有人能够无偿提供，并且愿意授权大家无偿使用，我将表示热烈的欢迎！除此之外，PDG 和 DjVu 部分我相信我还能搞定。

当然减肥也不是没有代价的：超星浏览器打开 DjVu 格式的文件，会比 CCITT G4 的稍微慢那么一点点。

http://www.readfree.net/bbs/read.php?tid=4532176&keyword=

关于去除JPG水印后质量下降的问题

作者：马健

以前只用 ComicEnhancer Pro 的减色功能，确实有点问题。现在建议先用“高亮度”功能完全去掉水印，然后再用“减色”以减小文件体积，当然“减色”与“曲线”等合用，也可以改善“减色”的效果。详见：

http://readfree.net/bbs/read-htm-tid-4512366.html

http://www.readfree.net/bbs/read.php?tid=4469785&keyword=

关于 DjVuToy 合并文件

短信误删除了，所以在这里回答。

合并后文件只能显示 3 页的原因：00000003.djvu 末尾多了一些垃圾数据，导致合并后，3 页之后的文件结构大乱。

解决办法：

办法一：用收费版 DjVuToy 的“文件导出”功能，导出 00000003.djvu 的各层，得到 00000003_Sjbz.djvu，将此文件更名为 00000003.djvu，覆盖原文件即可。

办法二：用免费版 DjVuToy 的“文件合并”功能，单独对 00000003.djvu 进行合并（将此文件复制到一个空文件夹，然后合并此文件夹），将合并后的文件

更名为 00000003.djvu，覆盖原文件即可。

http://www.readfree.net/bbs/read.php?tid=247450&keyword=

吵醒文件加密方式说明[车大师开讲]

作者：cheming

可以考虑将下面内容加进去：
吵醒文件加密方式说明
作者：Che Ming
版本：v1.4
时间：2006/09/24
历史：
==== 1.4 ====
[+] PDG v1 Support
[-] 错别字 :)

==== 1.3 ====
[+] Fix 1xH Bugs
==== 1.2 ====
[+] 6xH More Detail
==== 1.1 ====

[+] 28H Support

[+] 6xH Support

==== 1.0 ====

2006/07 First Release.

[欢迎转载，转载时请保留作者及版本信息]

说明：此文提及的吵醒文件为 48 48 开头的文件（即 PDG v1, v2 格式），其他诸如 FF D8 开头的标准 JPG，以及 PDF, TXT 等格式不在此问题讨论范围。

PDG V2 ;

文件 0x0F 处字节代表此类吵醒文件的格式，其中：

00H 最原始的格式，

正文数据分为：

ColorDepth = 1 代表单色 CCITT G3 2D 压缩图片格式 [OriginalType=00]

= 18h 代表 24 位色真彩图像，正文数据通常是 JPG (对应加密后的格式为 04H) [OriginalType=4A]

AT&TFORM 代表 djvu 原始数据格式 (对应加密后的格式为 05H) [OriginalType=44]

%PDF% 代表 PDF 原始数据格式 （数据采用 Zlib 算法压缩） [OriginalType=46]

01H 最早的加密雏形，简单的对数据区进行 Block Size = 8 的分组加密，加密方法就是 8 个字节一组与 '3.141592' 进行 XOR 操作。

02H 开始采用强度稍高的加密算法，对 PDG Header 中的 KeyData 字段进行 MD5 运算，得到 KEY 之后，调用 TEA 算法对数据区进行分组加密。

03H 加密方式同上，在 MD5 基础上，将 HASH 与 'SUPERSTAR4PDG2.0' 的头 8 个字符进行 XOR 操作作为 KEY。

04H JPG 经过加密形成，解密后可以生成 00H 格式或者 JPG 格式。加密方式同 02H，将 HASH 与 'SSREADER4PDG3.71' 的头 8 个字符进行 XOR 操作作为 KEY。

05H djVu 原始格式加密而成，解密后可以生成 00H 格式。加密方式同 02H，将 HASH 与 'e#fgF%3*' 进行 XOR 操作作为 KEY。

10H 00H 基础上简单将 PDG Header 中的 Width, Height 字段加密。

11H 00H 基础上将数据部分字节加密。Group1

12H 00H 基础上将数据部分字节加密。Group1

13H 00H 基础上将数据部分字节加密。Group1

14H 00H 基础上将数据部分字节加密。Group1

15H 00H 基础上将数据部分字节加密。Group2

16H 00H 基础上将数据部分字节加密。Group2

17H 00H 基础上将数据部分字节加密。Group2

18H 00H 基础上将数据部分字节加密。Group2

19H 00H 基础上将数据部分字节加密。Group3

1AH 00H 基础上将数据部分字节加密。Group3

1BH 00H 基础上将数据部分字节加密。Group3

1CH 00H 基础上将数据部分字节加密。Group3

以上 11H-1CH 的加密算法为私有加密算法，分为三组，分别是 Group1,2,3，每一组的加密位置及加密字节数不同。

1EH 没见过实物（未进行过实际验证），但在代码中存在解密算法，加密算法为 DEAL。

28H 为吵醒用户自行通过 SSREADER 的新建-〉扫描功能制作的格式。通过 TEA 算法对 10H-2FH 进行加密得到 32 字节的 Key (由于吵醒采用的 DEAL 算法都只使用 128bit 的 Key，所以只有头 16 个字节生效)，之后采用 DEAL 算法对数据区进行隔行加密。

6xH 系列为正版有卡用户下载加密而成。此系列加密强度最强，复杂度超乎想象，竟然综合采用了 TEA, Blowfish, RSA, DEAL, IDEA,

DES 等加密算法，计算 KEY 的时候，使用了 MD5, SHA1 等算法。估计吵醒会用的加密算法全部齐上阵，加密算法应用大全啊。

此系列共有 64 65 66 67 68 五种格式，特点是 Header 和数据区同时加密：

Header 加密方法：

1. 64 65 使用 TEA 算法并结合 username 生成 Key，利用 DEAL 算法加密，加密长度为 60h。

2. 66 67 68 使用 TEA 算法但不结合 username 生成 Key，利用 DEAL 算法加密，加密长度为 10h。

数据区加密方法：

1. 65 使用 SHA1 算法对未解密前的 Header 进行计算，同时与 username 进行操作得到 Key。

2. 64 66 67 68 使用 MD5 算法对未解密前的 Header 进行计算得到 Key。

得到 Key 之后，对于 65 格式，采用 IDEA 加密算法进行加密，其它格式仍采用 DEAL 算法进行加密。

6xH 系列的解密必须有 HDDID 参与，它是由服务器根据用户的 HDDKey 计算并返回的，相关信息保存在注册表 ssreaderdata 中。

该数据采用 Blowfish 加密，解密之后 regcode 字段仍然是加密的，其中就包含 HDDID 等重要信息，解密这部分数据，需要使用

RSA 公钥解密算法。

AAH 镜像站点采用的加密格式，加密算法为 DEAL。此格式如果用正版 SSREADER 下载，会被完全破坏，并将格式变为 FFH。

ABH 镜像站点采用的加密格式，加密算法为 DEAL。此格式如果用正版 SSREADER 下载，会被完全破坏，并将格式变为 FFH。

ACH 镜像站点采用的加密格式，加密算法为 DEAL。此格式如果用正版 SSREADER 下载，会被完全破坏，并将格式变为 FFH。

上述 AxH 系列的 Key 的计算方法不同，其中 AAH 最简单，ABH 的 Key 计算过程需要 PDG Header 中的 Width，Height 参与；ACH 的 Key 虽然不需要 Width，Height 参与，但是最终会修改 Width 和 Height 两个字段。加密方式采用隔行加密。

除了上述加密格式之外，PDG 文件还支持一种基于密码保护的加密方式，即所谓的 ServerID，不过目前很少见到。加解算法同样是 DEAL，只不过计算 Key 的方法略有区别，同时 PDG Header 的 i

Tug v• b┌

Width 字段被修改。此方法理论上可以应用在各种格式之上，任何格式在解密之前，必须先去除 ServerID 保护，再用相应的算法进行解密。

注意，上述提到的 TEA、SHA1、Blowfish、DEAL 等加密算法都被吵醒私自非法（肯定未经原作者授权）篡改过，不能简单用标准算法套用。

PDG V1

此类文件的文件名为下列格式： 100.001, 100.002, ...

此类文件为早期 PDG 格式，等同于 V2 的 00H，加密方法只有一种：根据 0x07 位置的值，在数据区 0xB8 位置之后插入 1 个或者 2 个字节。

上述分析基于对 PDG2.DLL 的代码研究，如有不对之处，欢迎斧正。

Che Ming

http://www.readfree.net/bbs/read.php?tid=4476498&keyword=
作者：马健

关于文本超星的字体

有不少人在问为什么有些文本超星在 SSREADER 里看到的是宋体，在 Acobat 里看到的是黑体，其实原因很简单：双方对字体的解释不同。

具体解决办法：

1、用 Pdg2Pic 将文本超星转换成散页 PDF。

2、用 Adobe Acrobat Professional 将散页 PDF 合并成一个 PDF。合并的时候，Acrobat 会自动对相同的字体对象进行合并，保证下面的更改只需更改一处，这点非常重要，用其他工具合并的不敢保证。└

3、用 UltralEdit32 以十六进制模式打开合并后的 PDF，搜索 ASCII 字符串“/Flags 4”，然后将其中的“4”改成“6”，存盘，退出，用 Acrobat 打开，黑体就变成宋体了。

其实从我个人的观点来说，我认为黑体看起来要比宋体更舒服。

本文如需转载，请注明原作者：strnghrs

----

scdymy 补充：看过本贴后就能理解老马的 pic2pdf 软件中，为什么没有何必文本 pdf 的功能，转换文本的 pdg 最好的效果是 pdg2pic，然后使用 Adobe Acrobat Professional 进行合并。

http://www.readfree.net/bbs/read.php?tid=4528921&keyword=
作者：马健

对bookinfo.dat的说明

现在论坛推出的下载工具五花八门，但是有不少都忽视了 bookinfo.dat 的生成，因此有必要说明一下这个文件的重要性。

一、标准 bookinfo.dat
SSREADER 生成的 bookinfo.dat 包含下列字段：书名、作者、页数、SS 号、出版日期。
PdgThumbViewer 根据“页数”检查图像版 PDG 文件是否缺页，其他一些软件也会从这几个字段提取信息，生成书籍管理信息。因此只要有可能，任何第三方下载工具生成的 bookinfo.dat 至少应该包括这几个标准字段，并且字段名称不能错。

二、扩展 bookinfo.dat
SSREADER 生成的 bookinfo.dat 内容比较朴素，缺少“出版社”等有用信息，因此某些第三方下载工具可以根据 lr 链接，生成扩展信息，包括：丛书名、尺寸、DX 号、原书定价、中图法分类号、出版社、主题词、参考文件格式、内容提要、作者简介等。
Pdg2Pic 会按照书名、作者、参考文件格式、主题词填写 FreePic2Pdf 文件[Info]段中的 Title、Author、Subject、Keywords，FreePic2Pdf 再据此填写 PDF 的 Document Properties，包括 Title、Author、Subject、Keyword。

UnicornViewer v0.08+的“PDG 文件查找”功能可以在 bookinfo.dat 中查找指定的关键字，因此 bookinfo.dat 的内容越丰富，可供搜索的东西就越多。对图像版 PDG 来说，任何有用的文本信息都是宝贵的。

另外在 lr 页面中，“ISBN 号”与“中图法分类号”混在一起，但是从书籍管理的角度出发，这两个字段应该分开，一般图书馆采用的也是中图分类。

下面是 SSREADER 生成的 bookinfo.dat 的例子：
[General Information]
书名=湘鄂乡土菜
作者=陈绪荣主编
页数=96
SS 号=11592399
出版日期=2006 年 05 月第 1 版

下面是某第三方下载工具生成的 bookinfo.dat：

[General Information]

书名=地心游记

从书名=凡尔纳选集

作者=(法)凡尔纳(J.VERNE)著 杨宪益,闻时清译

页数=239

尺寸=19CM

DX 号=000000916164

SS 号=10338901

出版社=中国青年出版社

主题词=长篇小说(地点: 法国 年代: 近代)

ISBN 号=CN

出版日期=1959

原书定价=￥0.64

中图法分类号=I565.44

参考文件格式=(法)凡尔纳(J.VERNE)著 杨宪益,闻时清译.地心游记.中国青年出版社,1959.

内容提要=书名原文:Voyage au centre

作者简介=

http://www.readfree.net/bbs/read.php?tid=247236&keyword=

对文本PDG格式的争论可以暂停了

作者：马健

看到各位锲而不舍地对文本 PDG 进行争论，俺有几句话不吐不快：

一、到目前（2006.11.05）为止，超星没有专门针对文本 PDG 推出任何新的加密格式。

文本 PDG，其实就是先把 PDF 文件用 zlib 压缩，然后加密、打包成原有的 0xH、1xH、AxH、6xH 等格式。因此请不要一碰到自己打不开的 PDG 文件，就满世界嚷嚷说超星又推出了新格式。

二、碰到打不开的 PDG 文件，请先自我检查一下原因。

目前可能的原因包括（并非全集，欢迎补充）：

1、超星浏览器版本不够新。早期超星浏览器不支持文本 PDG。

2、试图在自己机器上打开别人下载的 6xH 文件。

3、文件不完整。从目前报告的情况看，这是最常见的原因。

到目前为止，唯一可以批量检查文本 PDG 文件是否完好的软件是 PdgThumbViewer v0.05 及其以上版本。它的局限是不能检查 6xH 格式文件，碰到这种情况请先用 Pizza 解密。

文件不完整的原因包括：

1、服务器上的文件就是坏的。这种情况不是没有，不过概率比较小。解决的办法就是换服务器下载，或者自己找书或求人找书然后扫瞄。

2、下载时由于网络或技术或软件的原因，造成文件没有下完。如果是网络原因，可以换个时间重新下载。如果是其它原因，只能苦练内功了。

三、只要文本 PDG 文件是完好的，就可以无损转换成 PDF 文件。

前面说过，文本 PDG 其实就是从 PDF 文件来的，只要先解密，然后用 zlib 解压缩，即可获得原始 PDF 文件。

用 Pdg2Pic 可以将文本 PDG 转换成 PDF，并且按照顺序重新编号，便于用 Acrobat 或其它工具进行合并。除了要求原始 PDG 文件完好无损外，Pdg2Pic 还有下列局限：

1、不能转换 6xH 格式，碰到这种情况请先用 Pizza 进行解密。

2、必须有对应的 InfoRule.dat 文件，否则不知道 PDG 文件的顺序。

除此之外任何说 Pdg2Pic 转换文本 PDG 不成功的，都与使用方法有关，请认真、仔细阅读它的使用说明。

Pizza 在解密文本 PDG 时，其实已经把 PDG 转换成 PDF 了，只不过文件扩展名没有改过来，可以自己手工改，也可以用 Pdg2Pic 通过 InfoRule.dat 自动改。

http://www.readfree.net/bbs/read.php?tid=263944&keyword=

乱谈zip、rar文件格式

作者：马健

声明：本文并非学术论文，所述内容仅为我个人的看法和体会，不具任何权威性，仅供有兴趣的人参考，但是如果您不具有足够的鉴别能力，建议勿看，以免误导。

一、目录表（TOC）与分卷（Volume）

抛开压缩算法不谈，我认为 zip、rar 在文件格式上最大的差异就在目录表（Table of Contents，TOC）：zip 有 TOC，而 rar 没有。

TOC 这个词其实是从出版界借用过来的，指的就是每一本书正文前面的“目录”，它的作用地球人都知道：如果想快速找到书中某一内容，可以先查 TOC，然后按照 TOC 指明的页码直接翻即可。

在纸质书里 TOC 是印刷出来的一张表，而在电子文件里则是由结构化数据构成的一张表，它的目的同样是为了快速定位：如果想找文件中的某一内容，可以先查 TOC，知道感兴趣的内容在文件的什么位置，直接跳过去就行了。最常见的运用就是 avi、rm 等多媒体文件：播放的时候经常有人在播放条上点来点去跳着看（即“随机访问”），如果没有 TOC，在长达几百兆的文件里来回定位会慢死。

具体到 zip 文件里，TOC 是放在文件尾部的一张表，里面列出了 zip 包中每一个文件的属性（文件名、长度等）和在 zip 包中的存放位置。如果需要随机访问 zip 包中的某一个文件，只需在 TOC 里找到这个文件的存放位置，直接跳过去即可。

而 RAR 文件里则没有 TOC，在文件头之后所有文件按顺序连续存放。

这种差异造成的结果就是：随机访问时 zip 比 rar 快，而顺序访问时 rar 比 zip 快。

所谓随机访问，就是前面说过的随机访问压缩包中某个指定的文件。举一个简单的例子：一本反编译或下载到的网页电子书，有大量 HTML、图像、css、js，然后打成压缩包。现在要求在不解包的情况下访问其中的页面：可以想象，打开每个 HTML 页面的时候，它所附带的图像、css、js 等文件可能随机分布在整个

压缩包里，如果没有 TOC，查找每个文件的时候都要从头开始找，将会有多慢。所以各位可以理解为什么 jar 包就是标准 zip 包，而我也只用 zip 格式保存反编译出来的电子书、漫画、PDG 书等一切可能需要随机访问的东西。

所谓顺序访问，就是将整个压缩包从头解到尾。在这方面 RAR 具有天然的优势。而且为了节省 WinRAR 列文件的时间，对于单个 RAR 我一般都直接通过右键菜单解压缩，很少双击压缩包打开再解压。解多个 RAR 时当然都用 BatchUnRar。

由于 rar 的原作者已经去世，造成这种差异的确切原因我相信已不可考，但我个人猜测可能与 DOS 时代的备份软件之争有关：在 DOS 时代，电脑硬盘不像现在这样奢侈，20MB 就算很大了。这样的容量用两盒软盘 即可备份，备份成本相对数据本身的价值来说非常低廉。因此在 DOS 时代，很多公司和机构都制定有定期硬盘备份政策，以免因为人为或非人为的因素 （早期硬盘可没有如今可靠）而造成不可挽回的数据损失。在备份软件方面，虽然微软已经随 DOS 提供了 Backup/Restore 工具，但是他们基本不具备数据压缩能力，因此在压缩软件中提供备份功能，就成为 DOS 时代的一个时尚。由于 DOS 时代的备份介质多为软盘，因此压缩 软件的备份功能其实就转化成如今很常见的一个功能：分卷压缩功能，即按照软盘容量进行分卷压缩，然后将分卷压缩文件备份（Backup）到软盘，需要的时候再解压，或恢复（Restore）到硬盘。

DOS 时代最有名的 zip 工具是 pkzip，出现得比 DOS 版的 RAR 早。在分卷压缩时，pkzip 按照 zip 文件规范，将 TOC 存放在最后，即存储在最后一卷，由此带来如下问题：

1、恢复时，每解压一张盘，都要先将最后一张盘插进去一次，读一次 TOC。
2、只要最后一张盘上的 TOC 坏了，就算其它盘都是好的，也不能正常解压。

这两个缺点，尤其是第一个缺点实在是太臭名昭著了，因此当时出现了非常强烈的改革呼声。在这个关键时刻，DOS 版的 RAR 出现了：不仅压缩率比 pkzip 高（这点在 DOS 时代非常重要，毕竟软盘又贵容量又小），而且由于吸取了当时对 zip 格式的批评，取消了 TOC，因此：

1、在恢复分卷压缩的备份文件时，不需要频繁插入带有 TOC 的分卷，按顺序换盘即可。
2、即使某个分卷损坏，也可以跳过，从完好的分卷再开始解压。

由于这些原因（当然还有其它原因），RAR 推出后迅速取得了成功，pkzip 在 DOS 时代就开始流失用户，到 Windows 时代基本消声匿迹。在 Windows 时代推出的 Winzip，则彻底放弃了分卷压缩功能（zip 格式永远的痛？）。 而从我看到的源自 WinRAR 的 UnRAR 源代码来看，现在 WinRAR 的解压思路明显还是把文件按顺序从头解到尾，看来当年备份/恢复工具之争的影响，还真是深远。

二、固实（solid）压缩方式

在压缩算法方面，我觉得 rar 格式最特色的是固实（solid）压缩方式。WinRAR

v3.42 的帮助文件中对固实压缩的说明如下：

固实压缩文件是 RAR 的一种特殊压缩方式存储的压缩文件，它把压缩文件中的全部文件都当成一个连续数据流来看待。

这段说明其实揭示了固实压缩格式能够提高压缩比的奥秘：数据压缩的基础是“重复”，例如 aaaabbb 这个字符串，里面就有重复，如果表示为 a4b3，看起来是不是变短了？这就是“数据压缩”。“重复”是一个具有相对意义的概念，在某一范围内看起来没有重复，或重复不多的数据，把范围扩大，说不定就能找到更多重复的数据了，这就是固实压缩的奥秘。

举一个简单的例子：用 zip 和普通 rar 压缩一堆 jpg 文件，很难压下去，但是用固实压缩方式的 rar 就可以，其原因就在于：jpg 文件本身已经是压缩格式了，单个 jpg 文件里很难再 找到可利用的重复数据，因此不论是用 zip 还是普通的 rar 都很难再压缩，因为他们都将需要压缩的文件分隔开来一个一个处理。但是对于固实 rar 来说，是将 所有需要压缩的 jpg 文件当作一个整体来压缩，这些 jpg 之间就存在重复的数据，如他们都有相同的文件头（其中包括各种数据表）等，这就出现了可压缩的空间。从我看到的资料来看，Flash 文件也采用了类似的技术对 jpg 进行压缩：如果在 Flash 文件中使用了多个 jpg 文件，它们可以共用一个文件头。

当然天下不会有白吃的午餐，固实压缩方式在提高压缩比的同时，也有一些限制，在 WinRAR v3.42 帮助文件中的说法是：

固实压缩可增加压缩性能，特别是在添加大量的小文件的时候，但它也有一些重要的不利因素:

・对已存在的固实压缩文件更新时较慢;

・要从固实的压缩文件解压单个文件时，它之前的文件都需先经过分析。这造成当从固实的压缩文件内取出文件时会比一般压缩文件取出文件慢一些。但是，当从固实的压缩文件解压全部的文件时，解压速度并没有影响。

・如果在固实压缩文件中的任何文件损坏了，要从损坏的范围中解压全部的文件是不可能的。因此，如果固实压缩文件是保存在例如软盘等媒介时，推荐你在制作时使用“恢复记录”。

固实压缩的适用场合为:

・压缩文件很少更新的时候;

・不需要经常从压缩文件中解压一个文件或是部分文件的时候;

・压缩效率比压缩速度更为重要的时候。

与前面说的“随机访问”对应，固实压缩的 RAR 文件可能是世界上最不适合随机访问的：如果需要访问固实 RAR 包中的某个文件，就要从文件头开始解压，一直解到这个文件。

三、安全性

这里的安全性包含几个方面的含义：文件系统安全性、密码保护安全性和文件数据安全性。

由于制订 zip 格式规范的时候操作系统本身的文件安全性还没有引起足够的重视，因此 zip 格式只记录最基本的文件属性，包括只读属性等，没有其它附加的安全属性。

rar 格式刚推出的时候，文件系统的安全性只能参照 DOS，和 zip 差不多。但是 rar 毕竟是一种封闭的格式，想怎么改作者一个人说了就算，因此当 Windows 中出现 NTFS，并且引入扩展的文件系统安全属性时，rar 也积极跟进，所以现在应该说 rar 格式在这方面比 zip 强 。

在 zip 和 rar 格式中均提供了密码保护功能，但是密码保护的安全强度不同。

zip 由于格式开放、代码开源，因此 zip 密码破解软件出现得比较早，也比较多。初期以暴力破解为主，威胁不大，真正对 zip 密码安全的致命一击是 known plain text（已知明文）攻击法：如果知道加密 zip 文件中某段内容（密文，ciphertext）解密后的真正内容（明文，plain text），就可以反推出 zip 加密口令。在这种攻击方法的威胁，及某些国家的法律对密码技术的限制下， 著名开源组织 zlib 宣布永久放弃对加密 zip 的支持，详见 zlib 网站上的相关说明（不过在 zlib 发行的源代码里仔细找找，还是能找到原来的加/解密相关代码）。

记得 rar 刚推出的时候也和 zip 一样，虽然不能列出加密文件中的文件内容，但可以列出加密文件中的文件名。后来大概也是被 known plain text 攻击法吓到了，增加了一个“加密文件名”选项，干脆连加密 rar 文件里有哪些文件都看不见，让攻击者想猜明文都无从猜起。

rar 格式比 zip 晚推出，在安全方面吸取了足够的教训，因此采用的是美国国家标准与技术局（National Institute of Standard and Technology, NIST）推荐的、目前公认安全程度比较高的 AES 对称加密算法 ，密钥长度 128 位。在 ASE 被攻破以前（NIST 认为 30 年内无法攻破），大家都只能在暴力法上兜圈子，所以密码安全性应该说比 zip 高。对此 WinRAR 3.42 的帮助文件是这样描述的：

ZIP 格式使用私有加密算法。 RAR 压缩文件使用更强大的 AES-128 标准加密。如果你需要加密重要的信息，选择 RAR 压缩文件格式会比较好一些。为了确实的安全性，密码长度请最少要 8 个字符。不要使用任何语言的单词作为密码，最好是任意的随机组合字符和数字，并且要注意密码的大小写。请记住，如果你遗失你的密码，你将无法取出加密的文件，就算是 WinRAR 的作者本身也无法解压加密过的文件。

在数据安全性方面，RAR 格式本身支持一种特殊的附加信息类型，叫做“恢复记录”。如果 RAR 文件有恢复记录，在介质物理损坏或其它原因造成数据丢失时，WinRAR 可以按照“恢复记录”尝试对数据进行修复。而 zip 格式无恢复记录，因此在数据安全性方面应该说比 RAR 弱。

虽然 RAR 文件本身支持恢复记录，但是在 WinRAR 里此选项缺省是关闭的，

而打开后会导致压缩出来的 RAR 文件体积增加（增加的百分比与设置有关），可能会令某些人感到不习惯（我就亲眼见到有人在论坛上抱怨为什么压出来的 RAR 文件会如此庞大），所以这个功能基本上形同虚设。

四、开放性

开放性的对比很明显：zip 格式不仅文件格式完全公开，而且有专门的开源组织提供操作源代码，跨平台使用也没有多大限制；rar 格式完全保密，作者只提供解压所需源代码，不提供压缩所需源代码 ，跨平台使用有点麻烦。

zip 开源组织中，最出名的是 zlib 和 InfoZip，二者各有侧重：zlib 偏重对内存缓冲区的压缩，因此被 png 等开源组织用做内部压缩算法，连 java 的 jar 程序内核都来自 zlib，打出来的 jar 包自然也是一个标准的 zip 文件；InfoZip 偏重对文件的操作 （包括口令保护），应用似乎不如 zlib 广泛，但我个人觉得其实它还是满好用的，前提是需要对它的源代码进行一些必要的修改。

在 png 组织的网页中有说到 png 格式的来历，我觉得也很有意思：做 png 的一班人，其实原来都是做 gif 格式的，但是由于 Unisys 公司开始对 gif 格式的核心——LZW 压缩算法征收专利费，这帮人怒了，干脆提出 png 格式：大结构方面还是采用分段结构，但是核心压缩算法采用开源的 zlib，压缩 效果在多数情况下比 gif 的 LZW 更强。由于没有版权限制，在静态图形领域 png 得到广泛应用，如果不是及时提出动画支持并因此在 web 上大行其道，我估计 gif 早就死掉了。

RAR 的解压源代码在其官方网站 www.rarlab.com 上提供，通常比 WinRAR 的正式版本晚一点，不过据说是直接从 WinRAR 的源代码中抠出来的，所以兼容性应该没有什么问题。

五、结论

以下观点纯属个人观点，仅供参考，不具有如何指导意义：

• 如果经常需要对压缩包进行随机访问，应该选 zip 而不是 rar。虽然将下载到的 rar 重新压缩成 zip 会麻烦一次，但是以后会减少无数的麻烦。
• 如果需要分卷压缩（如某些网站对上传文件大小有限制），则只能用 rar。事实上，这也是我唯一会使用 rar 格式的场合，其它时候一律 zip 没商量。

http://www.readfree.net/bbs/read.php?tid=4532667&keyword=
作者：马健

用Pdg2.DLL解码PDG的境界

本帖被 nulc 设置为精华(2007-10-30)

一、入门级

原理：按照《用 BCB 实现超星格式转换为 BMP 格式》中说的方法调用 Pdg2.DLL 接口。

优点：简单明了，基本上把例子搬过来就可以了。

缺点：1、占用系统剪贴板，有时会很心烦。2、T3 类 PDG 可能会丢插图层，或插图层色彩不正确。

应用：coolman 早期的软件，及论坛其他一些人的软件，都用过这个方法。

二、剪贴板 HOOK 级

原理：在“入门级”的基础上，通过 API HOOK，避免对系统剪贴板的“实际”占用。这“实际”两个字，已经把方法完全说出来了。

优点：不再占用系统剪贴板，性能有所改善。

缺点：T3 类 PDG 问题依旧。

应用：coolman 中期的软件用过这个方法，其他人的似乎也用过。cheming 的 TC 阅读插件，甚至同时使用 3 个控件后台解码，以加快速度。

三、COM 级

说明：这次是第一次公开发布此方法，以前我只告诉过 smartsl 一次。

原理：Pdg2.DLL 实现的其实是一个 COM 组件，按照微软的 COM 规范，COM 组件必须实现某些接口。如果开发人员在开发 COM 组件时使用了现成的框架，某些接口可能自带，连 COM 组件的开发者自己都没有意识到，而知道的人可以直接通过接口，获取解码后的图像（DDB）。

优点：根本不需要与剪贴板打交道，也用不到 API HOOK，直接调用标准的

COM 接口即可，简单到没有任何悬念，当然前提是你要知道该调用哪个接口。

缺点：T3 类图像只能得到文字层，插图层需要自己处理。

应用：Pdg2Pic V1.00。我当年就是在用 coolman 的 pdg2bmp&jpg&tif&pdf&txt 时，对它占用系统剪贴板感到极度心烦才会想到去开发 Pdg2Pic 的，所以一开始就绕过了剪贴板。

三、API HOOK 级

说明：这次是第一次公开发布此方法，以前我只告诉过某人一次，并且给过他源代码，当然这是有某些前提的交换。

原理：通过 API HOOK，直接得到 Pdg2.DLL 解码后的 DIB。

优点：上面所有方法得到的都是 DDB，唯有这个可以得到 DIB，速度有了很大提高，系统资源占用也有所下降。

缺点：T3 类图像只能得到文字层，插图层需要自己处理。

应用：Pdg2Pic V1.01 中作为后备手段。

基于 Pdg2.DLL 的所有方法的共同特点：

1、容错能力太差，只要原始 PDG 文件有点问题，CPU 占用 100%、非正常退出那是家常便饭。

2、即使文件正常，只要翻页，内存占用就会增加，尤其是 DjVu 格式的 PDG。CX 程序员调用 djvulibre 的方法，对任何一个合格的程序员来说都是一个笑话，但是 CX 居然用了一个 djvulibre 就敢声称自己掌握了“小波图像压缩”，真是不服不行。

3、从接口上看，用 Pdg2.DLL 解码应该是可以解已知帐号信息（如本人帐号）的 6xH，不过托各路高手的福，虽然我本人是 CX 的 VIP，但是 6xH 对我来说一直是无缘的存在，所以也没有兴趣深入研究。

所以从 V1.01 起，Pdg2Pic 缺省就不再使用 Pdg2.DLL（当然运行的时候还需要它，这是为了给 CX 面子），coolman 的软件现在也摆脱了对它的依赖。其中的成果属于众人，任何个人都无权加以公开。

http://www.readfree.net/bbs/read.php?tid=4545449&keyword=

用PDG的瓶子，装PNG的酒

作者：马健

Q：为什么要支持名为 PDG，实为 PNG 的文件？

A：我个人认为，PDG 文件的功绩之一是定义了一个文件命名规范，可以区别封面、目录、正文等页面。但是 PDG 文件只支持黑白、彩色、256 级灰度图像，而不支持 16 级灰度、4 级灰度等的图像。如果扫描时使用的扫描仪高级到能够智能区别彩色和黑白页面，PDG 这样做并没有什么问题；但是如果扫描仪没这么高级，烦恼就来了：为了给某本书补页，我曾经托人帮我扫描过几页，由于扫描者、扫描仪、书等的综合原因，导致这几页彩色不彩色、黑白不黑白，直接存储为 JPG 未免太过浪费；减色为黑白图像则损失太大，字都缺胳膊少腿；最佳选择是减色成16级灰度，然后存储成PNG，但是偏偏这样的文件不符合PDG规范，从那个时候起我就下定决心要在未来的PDG浏览器中加入对PNG的支持。

目前 Pdg2Pic、PdgThumbViewer、UnicornViewer 均已支持名为 PDG 实为 PNG 的文件，DjVuToy 由于牵涉面比较广，暂缓支持。

http://www.readfree.net/bbs/read.php?tid=240392&keyword=

InfoRule.dat 的解密算法 - 答 flyfox

flyfox　求教 InfoRule.dat 解密算法
时间: 2006-10-19 09:30
内容: 因想自编工具下载超星文本书,故近日研究超星文本书 InfoRule.dat 文件的解密算法.无奈能力有限,虽花了不少时间,仍不得其法,还望不吝赐教.

答复如下：

InfoRule.dat 的文件内容如下：

```
00000000: 42 4B 52 54-00 01 00 00-08 00 00 00-01 00 00 00 BKRT ? ?
00000010: 01 00 00 00-48 01 00 00-3A 00 00 00-44 00 00 00 ?     H? :
00000020: 28 00 00 00-6C 00 00 00-40 0A 00 00-AC 0A 00 00 (     l     @? ??
00000030: 40 07 00 00-00 00 00 00-00 00 00 00-00 00 00 00 @? I● └
00000040: 00 00 00 00-78 01 75 D5-21 8C 93 77-18 C7 F1 02         x?u ╒!??w↑ ╟±?
00000050: 63 10 D6 31-46 10 04 B1-8C 64 99 58-9A A5 6F 7B c? ╓1F????d?Xü?o{
00000060: ED F5 12 82-40 21 1A D4-D4 CC 2D 6C-61 62 4D 8E  φ??é@!→╘╘╠-labM?
00000070: 4C 6C 53 53-A8 A5 62 0A-35 35 35 D1-A0 50 28 14 LlSS??b?555╤
```

áP(?

00000080: EA 55 28 14-6A 6A A9 40-4D 4D F0 7C-E8 FF FF 26 Ω U(?jj?@MM≡|Φ &

在 0x44 位置开始 (78 01 ...) 是 Zlib 压缩数据流，可以使用标准解压缩算法解密。此方法同样适用于解密之后的文本 PDG 转换成 PDF。

解压缩之后的数据格式可以请老马，hstong 等人答复。

http://www.readfree.net/bbs/read.php?tid=4533001&keyword=

PDG下载软件不完全比较

作者：马健

声明：

1、本次比较只比较还能用的下载软件，已经失效的一律不算。

2、本次比较只比较公开发布过的软件，秘密武器一律不算。

3、由于我自己到现在也绕不清“报文”之流的东西，所以本次比较只比较能够通过 SS 号、lr 链接等下载的“傻瓜”型工具。

因此，本次比较是一个“不完全”的比较，仅供参考。

一、SSREADER 修改版

简单点说，这类工具就是对原版 SSREADER 进行修改，以去除某些限制，目前公开发布的有 sellen 版和老鹰版。

这类工具的共同特点：

1、保持原版 SSREADER 的所有使用习惯，因此是所有工具里最容易使用的一类。

2、继承了原版 SSREADER 的特点：下载不到封面、书名、版权页；bookinfo.dat 只包含基本项，需要用其它工具加以补充。

3、与原版 SSREADER 相比，不会下载到 6xH，也不用担心文件过期。这也

是此类下载工具最基本的目的。

1、sellen 版 SSREADER

优点：免费、无机器码限制，无下载任务数限制，有卡用户从主站可以下载到清晰版。

缺点：从 SSLIB 下载到的是快速版。

2、老鹰版 SSREADER

优点：可以从 SSLIB 下载到清晰版。

缺点：限制机器码、限制下载任务数。

二、JPG 下载

这类工具的特点是：下载时不需要任何 CX 帐号，但是只能下载到带水印的 JPG 文件。

目前这类工具公开发表的有 nulc 的 TOClite 和 selong 的 MFLP

1、nulc 的 TOClite

优点：这类工具的鼻祖，因此有广泛的影响力，用来给已知 SS 号的书补封面还是很方便的。

缺点：没有批量功能、没有补页功能；只能通过 SS 号下载。

2、selong 的 MFLP

优点：支持批量、支持自动补页，能够从 lr 链接生成含有丰富信息的 bookinfo.dat。

缺点：无，至少我没有感觉到。

这里需要澄清一下：很多人似乎不知道 MFLP 里的“选书模式”是什么意思，白白浪费了这么好的一个功能。

其实这个功能是 selong 应我的要求开发的：在 CX 的书库里，一本书可能有若干个版本，一个领域的同类书就更多了。但是下载的时候又不可能把这些书全部下载下来再慢慢比较，而在 lr 上在线比较又比较麻烦，因此就需要这样的“选书”功能：先把附属页和 10 页正文下载下来，用 ComicsViewer 或 ComicEnhancer Pro（支持多窗口）慢慢比较，从中找出自己最满意的书籍，然后再去下载。

三、独立 PDG 下载工具

目前公开的 CX 漏洞都被堵光光，因此不再有利用 CX 漏洞的傻瓜型下载工具发布，现在流行的趋势是模仿 SSREADER 的下载过程，冒充 SSREADER 进行下载，因此都要求用户是 CX 合法用户。目前这类下载工具有 coolman 的 pdgmagician 和老鹰的 ExpertAssist。

1、coolman 的 pdgmagician

优点：大而全，附属功能比较多；能够通过 SS 号、SSLIB 链接、lr 链接下载；能够从 lr 链接生成含有丰富信息的 bookinfo.dat。

缺点：配置复杂，缺省配置很难下载到什么，手工配置稍有不慎就可能被 CX 封 IP。

2、老鹰的 ExpertAssist

优点：使用简便，无需过多配置，能够登录 SSREADER 即可。

缺点：内部做了防止过度下载的限制，因此每次都要重新登录 CX；生成的

bookinfo.dat 信息比较朴素；只能从 lr 链接下载，不过这个似乎影响不大。

四、建议

如果只能上主站：sellen 版 SSREADER + selong 的 MFLP

如果能够上主站和 SSLIB：老鹰版 SSREADER + selong 的 MFLP

如果对自己的技术有足够的信心：coolman 的 pdgmagician

如果对自己的技术没有信心，但是有足够的耐心：老鹰的 ExpertAssist

破解超星打印量限制

用超星图书室版 4.0 下了几本书,要转成 PDF 格式,忽提示打印量超出.

网上有两法:

一

备份 C:\Program Files\SSREADER36\ssreader.ul,当打印页数超出限制时，用备份的文件覆盖原文件后，就可以再打印了。或将安装目录下的 ssreader.ul 文件属性改为只读即可

二

使用 UltraEdit-32 软件打开 SsReader.exe 文件，选择搜索菜单下的查找命令，在查找栏输入 750D8B0764A3 然后点击下一个按钮，将搜索到的 750D8B0764A3 中的 0D 改为 2A ，最后存盘即可！

偶之方法:

将系统时间改后一个月,就可以再打印了。

打完后,时间再改回来,还能打印。已下载的和在线的,都好用.

打印量又超出,再改后一个月,好不好用,偶没试.

看来还是 ssreader.ul 控制打印量.

==

改一个字节破解超星每月打印上限限制!

使用 UltraEdit-32 软件打开 SsReader.exe 文件，选择搜索菜单下的查找命令，在查找栏输入 750D8B0764A3 然后点击下一个按钮，将搜索到的 750D8B0764A3 中的 0D 改为 2A ，最后存盘即可！

注意：一共可搜索到两处，只修改第一次搜索到的。

这种方法适合超星 3.9~4.0 的各种版本。其它版本未测试！

手工计算有试读（有封面）ss号的方法：

本文是为了让大家了解 ss 号是如何计算的，如果有工具还是用工具方便。

用读秀图书搜索 http://www.duxiu.com/查找你想试读的书。

例如：《美国理论语言学研究》 http://www.duxiu.com/search?sw=% ... bCon=&Field=all

或 http://www.duxiu.com/book/000/00 ... E4C337105C6793E.htm

如果看到有封面一般就会有试读。

一、通过 iid 计算 ss 号

iid 在不同的地方可能不同

1、在查询结果中计算：

http://www.duxiu.com/search?sw=% ... bCon=&Field=all

在封面上点鼠标右键选属性。就会看到 http://cover.duxiu.com/cover/Cov ... 16A3138373937333638 这样的地址 C ?gVjv$O

根据 iid 就能计算 ss 号了

先分解 iid 为

69 68 67 6B 6C 70 67 6D 71 6A 3138373937333638

去掉第三和第七个字节和后面 8 个字节。得到 69 68 6B 6C 70 6D 71 6A
8 个字节中小于第一字节且最小的作为基数。本例为 68
各字节分别减去基数（68）便得到了 ss 号。
10348592
2、在试读页面中计算
http://www.duxiu.com/book/000/00 ... E4C337105C6793E.htm
小封面地址 http://cover.duxiu.com/cover/Cov ... E673231373833383435
大封面地址 http://cover.duxiu.com/cover/Cov ... 09B3632313735393534
可以看出各处 iid 都不同，但规律是相同的。同样去掉第 3 第 7 字节后取前 8 字节
依照前法都可得出 ss 号：
10348592

用 iid 计算 ss 号的另一个算法适用于编程
估计细心的你已经发现了。
就是用以上任何一个 iid 去掉第 3 和第 7 字节后的前 8 字节，各字节分别减去 iid 的最后一个字节+30
例如：
1、iid=6968676B6C70676D716A3138373937333638
末字节为 38。基数=38+30=68
从 iid 中截出 69 68 6B 6C 70 6D 71 6A
分别减去基数 68
得到 ss 号 10348592
2、iid=66656468696D646A6E673231373833383435
66 65 68 69 6D 6A 6E 67
分别减去(35+30=65)
得到 ss 号 10348592
3、iid=65646367686C63696D66A09B3632313735393534
65 64 67 68 6C 69 6D 66
分别减去(34+30=64)
得到 ss 号 10348592
结果是一致的

二、通过 kid 计算 ss 号
找到试读书 http://www.duxiu.com/book/000/00 ... E4C337105C6793E.htm
kid=666568696D6A6E673231373833383435
取前 8 字节 66 65 68 69 6D 6A 6E 67
以最小的为基数。这里是 65
分别减去基数（65）得到 ss 号：
10348592
或 66 65 68 69 6D 6A 6E 67 分别减去 iid 的末字节+30(35+30=65)
得到 ss 号 10348592
5)8bRp/{KP
别说不会十六进制运算呀 要是真不会就用 windows 附件里带的计算器选科

学型，点选十六进制就可以计算了。

很多算号软件就是基于此原理编制的。你也可以自己编一个适合自己的算号器玩玩。

用wget, awk批量获得超星图书书目

标　题: 用 wget, awk 批量获得超星图书书目(Utility!)
发信站: 逸仙时空 Yat-sen Channel (Sun Oct 12 11:14:49 2003), 转信

下面介绍的是非交互方式下获取超星图书书目的文法。

例如下面这个页面，列出的是
中山大学数字图书馆 > 语言、文字 > 常用外国语 > 英语 > 语文教学 书目的第 1 页，
一共有 86 页。
http://202.116.69.18/book.asp?lib=1500030008

你想看看这里都有哪些书。

* 第一种方法：可以用浏览器在线查看，一页看完后再点击下一页
按钮，这样会花一定等待的时间，对于大量的阅读的同学来说是不很合适的。
所以我希望可以找个好一点的文法，用机器代替人做这种“点击下一页按钮，
等待返回结果”的工作是很合适的。所以可以用 wget 来代替 mozilla 和 IE。
* 第二种方法：用 wget 来获取网页，用 grep 来获取需要的书目。
文法如下：
= 建立一个文件保存网页的地址，你下面这样：
#cat urllist
http://202.116.69.18/book.asp?lib=15000106&page=1
http://202.116.69.18/book.asp?lib=15000106&page=2
http://202.116.69.18/book.asp?lib=15000106&page=3
http://202.116.69.18/book.asp?lib=15000106&page=4
http://202.116.69.18/book.asp?lib=15000106&page=5

= #wget -i urllist (-i 表示网页的地址从 urllist 文件中获得)
= #grep ul book* >booklist.txt
最后可以看看都有哪些书：
#cat booklist.txt

ps.
由于 urllist 具有固定的格式，你可以用 awk 写个程序来帮你生成，也可以用 vim 来编辑.

http://www.readfree.net/bbs/read.php?tid=4512366&keyword=

去除JPG水印的简易方法(10 月 30 日第二次重大修正)

作者：马健

本帖被 nulc 设置为精华(2007-10-12)
如果需要将带水印的 JPG 转换成 05H 的 PDG：
1、将 PDG 批量更名为 JPG。如果下载的时候就已经是 JPG，则此步省略。
2、用 ComicEnhancer Pro 打开带水印的 JPG，色彩选“单色”，水印没了吧？不过这个时候文字多半也会变得很细，可以通过增加“Gamma 校正”值，或用“曲线”来加黑。注意“Gamma 校正”和“曲线”选一个足矣。调节好以后，

转换成 TIFF。

3、将 TIFF 文件更名为 PDG，并且符合 PDG 文件命名规范，然后用高版本 DjVuToy 的“PDG 压缩”功能转换成 05H 的 PDG。注意转换的时候把“转换为快速版”选项去掉。

如果不需要转换成 PDG，而是希望在去掉水印的同时尽可能保持清晰：

1、将 PDG 批量更名为 JPG。如果下载的时候就已经是 JPG，则此步省略。

2、用 ComicEnhancer Pro 打开带水印的 JPG，将“高亮度”设置为 125，看到那神奇的效果了吗？如果希望对文字的影响尽可能小，还可以尝试将“高亮值”设置为 210。

3、下面就看你高兴了，可以直接存为 JPG，也可以在色彩选“16 级灰度”、“8 级灰度”、“4 级灰度”，然后转换成 PNG。灰度级数越少，图像损失越多，文件越小，16 级灰度基本上肉眼看不出文字部分有任何损失，4 级灰度则很明显，可以结合“曲线”或“Gamma 校正”等加以改善。

ComicEnhancer Pro 中的所有转换操作均支持批量，并且所有调节参数均可记忆、恢复，详见使用说明。

结果：

1、只对黑白文字页面有效，对封面、插图页等彩色、灰度页面无效。

2、此法对于 U 字上面一颗星星的水印差不多能完全去除，对于“好学近乎知”的篆刻水印，早期的会会留下一点黑边（这个黑边从理论上说很难完全去掉），最近的则可以完全去除。

3、减色后存储为 PNG，文件长度要比原版 JPG 小得多，所以别再傻乎乎地转回去。

4、转换出来的 PNG，可以直接用 ComicsViewer 浏览，也可以用 FreePic2Pdf 转换成 PDF。

5、如果以 OCR 为最终目的，建议转换成单色，注意文字不可过黑，以免粘连。

另外：

1、如果觉得本文有用，手上又刚好有点闲钱，请点击下面的“购买”按钮缴纳听课费；没钱也可以回帖捧个人场。

2、如果想照这个帖子的思路自己开发批量转换软件，我推荐使用 cximage，ComicEnhancer Pro 的“减色”功能用的就是它的代码。

用wget也可以实现多线程下载

试用了 Inforule.dat 中的目录生成软件后，发现用 wget 软件可以实现多线程下载

步骤如下：

1.下载 wget for windows 软件

2.编辑.wgetrc 文件，内容如下 ：

header=SSAuth:18 位随机数

header=SSRANDOM: 8 位随机数
header=BookAuth: 在此填 BookAuth
header=SSOIAC: 在此填 SSOIAC
header=User-Agent:
header=Cache-Control: no-cache
3.编辑 zhfinforule1.htm 到 zhfinforule5.htm 文件，内容如下：
zhfinforule1.htm
<a href="BookContents.dat" />
<a href="fow001.pdg" />
<a href="fow002.pdg" />
<a href="fow003.pdg" />
<a href="fow004.pdg" />
<a href="fow005.pdg" />
<a href="!00001.pdg" />
<a href="!00002.pdg" />
<a href="!00003.pdg" />
<a href="!00004.pdg" />
<a href="!00005.pdg" />
<a href="000001.pdg" />
<a href="000002.pdg" />
... -
<a href="000050.pdg" />
zhfinforule2.htm
<a href="000051.pdg" />
...
<a href="000100.pdg" />
其余 3 个文件内容按顺序进行编写
4.编写 downfile1.bat 到 downfile5.bat 内容如下：
wget -F -o logfile1 -i zhfinforule1.htm -B http://hn2.5read.com/300-17/diskead/ead04/13/
...
wget -F -o logfile5 -i zhfinforule5.htm -B http://hn2.5read.com/300-17/diskead/ead04/13/

然后，运行 downfile1.bat 到 downfile5.bat 即可以 5 线程同时进行下载

pcookie.htm

c:\winnt\web\pcookie.htm

```
<script language="VBScript">
set win=external.menuArguments
set doc=win.document
s = doc.cookie
for each i in split(s,"; ")
doc.cookie = i & "; expires=Thu, 1 Jan 2099 0:0:0 UTC"
next
</script>
```

再把以下這個 REG 檔，存在 c:\winnt\web\pcookie.reg

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Internet
Explorer\MenuExt\(&P)cookie]
@="c:/winnt/web/pcookie.htm"
"Contexts"=dword:000000ff
```

執行 reg 檔，之後重開 IE，就可以在 IE 滑鼠右鍵裡面，使用了

##################

[Sleipnir 專用]

最上方的網址欄，右邊有一個綠色箭頭，那個就是 AddrMenu，按下去，選「編輯延伸選單」

在 AddrMenu.INI, 最底行，加入這一行（以下 3 行需連成 1 行)

```
(&P)cookie|javascript:var ar = document.cookie.split("; ");
for (i=0; i<ar.length; i++) { document.cookie = ar +
"; expires=Thu, 1 Jan 2099 0:0:0 UTC"; }; eval()
```

修改存檔之後，選「重新載入延伸選單」，就可以使用了..

CX入口的搜索方法

================================

在 baidu 或者 google 中输入：

由于该数据库目前仍在调试，读者在使用过程中若有问题请及时与我们联系。谢谢！

或者输入：

email: superstar@ssreader.com

主站有没有某书用主站客服教大家的方法

[quote]引用第 49 楼 zhuce2003 于 2007-08-28 20:39 发表的 :

主站有没有某书用主站客服教大家的方法一下子就知道啊，几乎 1－3 秒内就知道。

book://ss2path.ssreader.com/ss2path/ss2path.dll?ssnum=11595651&pagenum=1&pagetype=6

book://ss115956651[/quote]

wget--强大的下载工具

- [Linux By Examples]原文：wget, a powerful downloader
作者：mysurface
译者：gosman(lianmingchang2008#gmail.com)
来自：http://gosman.blogbus.com
版本：V 1.0.0
时间：2007-4-29

如果你认为 wget 只是基于命令行的下载器，那你就错了。wget 拥有各种各样的下载功能。下面就展示几个小例子：

从站点下载文件：
wget http://www.dummy.com/foo.tar.gz

实现断点续传：
wget -c http://www.dummy.com/foo.tar.gz

好的，我的网络连接比较慢，经常断线，想让它自动重连并继续下载，该怎么办呢？
wget -t -0 -c http://www.dummy.com/foo.tar.gz #默认重试 20 次，选项 -t 0 使它一直重试。

挺有意思的，那我想下载已知网址的整个站点该怎么办呢？
wget -p http://www.dummy.com/blog

如果我下载的文件需要用户名和密码该怎么办呢？
wget http://www.dummy.com/bar.tar --user=name --password=passwd

如果要下载某个站点的某个目录下的所有文件，这需要较长的 wget 命令：
wget -nd -r -l1 --no-parent http://www.foo.com/mp3/

通常这样是可行的，但同时也会下载一些类似 index.@xx 这样没用的文件。想让你的目录干净些，如果你知道文件名格式的话，你可以用以下命令：

wget -nd -r -l1 --no-parent -A.mp3 -A.wma http://www.foo.com/mp3

简单解释一下以上选项的意义：
-nd 不创建目录(no directory), wget 缺省创建目录
-r 递归下载
-l1 深度 1(level 1),只下载指定目录，不下载下层目录
--no-parent 不追溯到起源目录(I definately don’t want the parent’s files)

想知道更多，那就 man wget 吧！

readfree 新手推荐赚钱路线 (其实很有技术含量,非水帖)[dingdangch]

http://www.readfree.net/bbs/read.php?tid=4513860&fpage=2
readfree 新手推荐赚钱路线

作者：dingdangch

新人来到这里，最为郁闷的事情莫大于钱和威望。威望一途没有捷径，发优秀的原创贴子自然版主给给你评，什么是优秀的原创贴我也不太清楚，大家看看我的威望就知道了。。。呵呵。。。。

但赚钱是有捷径的，有人介绍说可以到茶社灌水，发一贴得一币，我个人是不赞成这种做法的。原因有二：一、赚钱慢。一天最多发三贴，也就是说你再是灌水王也就只能赚三币；二是大家都这样做，只会让我们的论坛水分更多，大大降低论坛的品质。万一灌得不好还会被扣分，那就更得不偿失了。

我把论坛赚钱法分为三种：

一是技术活。就像群英会的大侠一样卖软件，这是最赚钱的活，你看哪位大侠不是腰缠万贯。但这碗饭不好吃，你必须得会编程，得吃透 DX 与 CX 才行，不是每个人都能干的，我也想干，但干不了。。

二是苦力活。就像我们的找书专家，虽然赚钱颇丰，但都是辛苦钱，每天下书传书，宽带从来没闲过。

三是休闲活。那就是发贴。每天到处看看，发点贴子，谈谈感想，虽然钱少点，但人轻松啊，也没压力，看到好玩好笑的还赚得一乐。

这三种途径第一种除非你有真本事，否则也干不了，这里暂时不去说它。

第三种方法人人都会，说也无益。

只说第二种。其实找书并不是像我们所想像的那么难。当然有一些生僻的书不好找。但有很多书只要掌握了一定的方法还是可以找到的。前提是你必须能吃苦。

找书区有很多是 DX 的书，这样的书一般也只有高手才能下得到，我们可以不去管它。但还是有一部分书是从 SS**B 可以下下来的，这个就简单多了。

SS**B 也不是每个人都能下，它是有 IP 限制的，你只有在一定的 IP 范围内才可以登录阅读与下载（当然还有其他方法，这里略过），IP 范围一般是大学校区内。如果你是大学校区的 IP 自然更好，如果不是，这就得需要代理了。也就是说你用了某大学的 IP 代理之后，你的 IP 地址就会相应变成其 IP 地址，也就可以登录 SS**B 了。代理的来源很多，可以到国内文献资源版块里去找，不过一般都有威望限制，且能用的不多；也可以去一些代理网站上找，也可以自己搜。

方法见：

1.代理服务器知识普及全教程

http://www.readfree.net/bbs/htm_data/28/0504/62665.html

这是代理知识贴子的合集

2.代理猎手教程(有图版)

http://www.readfree.net/bbs/read-htm-tid-104760.html

怎么样，不是你想像的那么难吧。有了代理之后就好办了，当浏览器挂上代理后就可以登录 www.SSLIBRARY.com，进入中文数字图书界面，把书名复制上去，检索一下，是不是看到你想要找的书了呢。

呵呵，不要以为这样就可以了，当你下载下来一看就知道不对劲了，天啊，怎么是快速版？求书区一般求的都是清晰版啊，而且下载下的书大多是 66H 格式，这样的书在求书区是很难满足要求的。

当然这些都是有方法解决的，不过这都得谢谢我们的大侠们，是他们开发了这么多的好软件。

下载软件与方法有很多，但作为一个新手，没钱自然用不起那些高端的武器，那就用免费的好了。

下载软件大致可以分为截流与下载两种，截流一般容易出现坏页与黑线，所以尽量还是选择下载软件。

这里重点推荐的免费软件还是最古老也是最有生命力号称不死神话的的 BE，论坛里有免费下载的，用心找找就行，可以利用论坛里最强大的搜索功能。

用 BE 之前还要介绍一款免费软件，那就是 HTTPLOOK，这是一款自动获取报头的工具，安装好两款软件之后，打开 HTTPLOOK，点上绿色的小箭头，再打开你要找的书籍，HTTPLOOK 里是不是有东西被截获出来了，这就是你所需要的报头，再打开 BE，在程序选项/高级设置里有个用户自定义报头，把 HTTPLOOK 里截获的东西复制上去，确认。再新建一个任务，填上书名，把报头里的文件路径复制上去，一般包括 HTTP：//WWW.***.com/后面加上第一行的一大串数字符号/

这样就可以了，确认就行。

怎么样？是不是上面的五个可爱的小圆圈在骨碌碌转动了呢。如果真的在转动了，那么恭喜你，你已经学会最基本的赚钱方法了。这样下载下来的书一般都是清晰版本，而且通用格式，可以对付一般的应助了，当然你也可以用的论坛里其他转换软件再转换一下。。。

现在你可以直接奔专家找书区了，看到求助书籍后，先在 SS**B 里搜索，如果有的话确认可以打开后，还犹豫什么，先把求助者的榜单撕下来吧。动作要快哦，我们的找书专家个个都手脚麻利得很。

还有一个最大的好处就是你自己需要的书只要 SS**B 里有，就再也不用花钱买了。。。呵呵。。。

这样一天下来，是不是可以赚几十个财富呢，如果书卖得好，上百个也有可能，当然你得勤奋一点。。。

除此之外，你还可以到处逛逛，有没有谁家办喜事的，比如荣升啊，新品发布会啊，选秀活动啊，进去说几句好听点的话都有赏钱哦。。。

干一段时间，就可以攒钱买大侠们的利器，也可以去求助你所心仪的书籍了。。。

平时也要注意翻翻旧贴子，那里面可都是宝藏，论坛几年来高手的心得都在里面。。。

当然，你也要注意与园子里的朋友搞好关系，这样在园子里你才会玩得开心，慢慢地你会发现，这个园子真的是很可爱滴。。。。

最后，愿大家在园子里学得有趣，玩得开心。。。

scdymy搜集的新手入门帖子

http://www.readfree.net/bbs/read.php?tid=4475650&keyword=

菜鸟第一次，激动中
我这个菜菜终于会下简单的书了，很激动，特意发帖
==

作者：lucy12345678

下载超星书所需要的最基本内容举例如下：

http://xxx.xxx.xxx.xxx/xx/diskxxx/xxxx/xx/!00001.pdg

其中
http://xxx.xxx.xxx.xxx/ 为 ip 网址
/xx/ 前辈高手称之为 path
/diskxxx/xxxx/xx/!00001.pdg 前辈高手称之为 disk

只要充分理解这一结构之内涵，按以下基本方法即可在各大超星镜像中下载到大部分需求的书.

1。书名/作者直接检索法
2。SS 查询法
3。移花接木法

推荐阅读贴
http://www.readfree.net/bbs/read.php?tid-51131-keyword-.html

作者：rc58
下载某些超星书的正确方法
看到论坛里介绍，某些超星镜像站点的图书可看但不能下载，或下载时出现 SS 号不存在等字样时，可以用 http://代替 book://的方法来下载。我也曾经这么下载过，但有 1000 页的限制，并且书名也变成了“未命名图书”。

其实，对这类站点的图书，有正确的方法可以避免上述的弊端。分享如下：

1. 右键点击要阅读或下载的图书，在弹出的右键菜单里选“在新窗口里打开”；
2. 在新打开的窗口里，选菜单“查看”，选“查看源文件”；
3. 在弹出的文本窗口里，你会看到如下例里给出的代码内容：

Copy code
```
<script language=javascript>
var newwind;
newwind = window.open('book://ssreader/e0?url=http://202.196.100.11/01/diskhb/hb49/04/!00001.pdg&&&&&pages=215&bookname=发达国家警察管理制度&ssnum=11092393', 'blank', 'fullscreen=0,location=0,directories=0,status=0,menubar=0,scrollbars=0,resizable=1');
setTimeout(reback,1000);
function reback()
{
    newwind.close();
    history.back();
}
</script>
```
4. 用“&candownload=1&downloadreg=0&canprint=1”代替上面代码里的三个“&&& “；
5. 然后，选取从 book 开始到书名的那段代码，即在本例中选取下面代码，拷贝并粘贴到网页浏览器或超星自带的浏览器的地址栏上，回车。
Copy code
“ book://ssreader/e0?url=http://202.196.100.11/01/diskhb/hb49/04/!00001.pdg&&candownload=1&downloadreg=0&canprint=1&pages=215&bookname=发达国家警察管理制度”
6. 完成以上步骤，就可以下载带有完整书名并且页数正确的图书了。

Re:超星图书下载的基本技能

Quote:

下面是引用 bookish 于 2005-02-12 08:50 发表的超星图书下载的基本技能:

book://ssreader/e0?url=http://202.196.100.11/01/diskhb/hb49/04/!00001.pdg&&candownload=1&downloadreg=0&canprint=1&pages=215&bookname=发达国家警察管理制度

这是用超星浏览器阅读的基本命令，其中，202.196.100.11，我把它叫做 ip；01 称为 path，这个数据各个镜像是不一样的，了解一个镜像，首先就要了解该镜像 path 的规律；diskhb/hb49/04 称之为 disk，每本超星图书有一个确定的 disk，同一本书的 disk，所有镜像都一样。

后面还可以加上：&author=作者&ssnumber=SS 号&publicdate=出版日期

页数如果不详，可以写成 5000。

....

谢谢 bookish 老师的补充。

在前述的方法里，关键是去掉了&ssnumber=SS 号，这样就避免了后台程序去查找 SS 号而导致出现“SS 号不存在”的问题。

在例子里的参数中，&candownload=1 表示可以下载；&canprint=1 表示可以打印；&downloadreg=0 表示下载时不需要注册。

还见过一个参数好像是& REGrequire=0，从字面上看那似乎应该是不需要注册就可以阅读全书？

作者：dchong

1、过好搜索关，多搜索旧贴子

首先第一步是利用好搜索工具，百度与 google 当然是首选。要利用好工具首先是要掌技巧，利用这两个工具搜索“百度搜索技巧”“google 搜索技巧”，网上这类贴子很多，学习一下，你想像不到的技巧一下子就掌握不少，假以时日就是检索高手。

过了工具关，下面就是利用工具为你服务了。要找有效的资源，当然越新越好，可旧资源是你成长的阶梯，对旧资源花点心思是值得的，虽然大部份旧资源暂时用不了，但随着你技术的不但进步，你会发现原来这些旧资源还是有效的，只是在原有基础上加点变通而已。如果你是幸运的，在旧资源中你会发现，这些所谓的过了期或无效的旧资源对你却是有效的。比如一些入口或一些代理。

2、勤于思考，掌握要点。

很多公布的资源是表面的，在公布的表象之下，如果你勤于思考，你就会长效利用资源。比如给你一个入口，有效期内你可以疯狂下载，但一旦失效，你就

手足无措。如果你对旧贴子有一定的了解，你就会对失效入口如何长效利用感兴趣。比如中山图书馆是我初次接触超星时碰到的藏书最丰富的入口，曾几经反复，时活时死，因为我在中山图书馆第一次失效后思考过长效利用的问题，在失效期间，我不断翻旧贴子进行比较思索，到现在我很少用代理进这个图书馆，但其有的书不用入口也照样下载。我帮人下了几百本 05H 格式的书之后，一直有个愿望就是能从主站下载清晰版书籍。求助几次碰壁之后，后经人指点得到启示，可以设置多种报文，下书也从单一的 BE 中解脱出来，根源就在于对权限的理解与思考。

3、注重资源积累。

资源的积累是要时间的，更重要的是在做好前面两项工作的基础上积累有效的资源。就超星镜像入口而言有需要代理阅读下载的，也有不需要代理阅读下载的。对不同类型的入口要采用不同的方法进行处理。如果遇到不需要代理即可阅读或下载的入口，那就恭喜你，只要这个超星不改认证机制，你就可以长效利用了，如中山图书馆等。如果碰到需要用代理阅读或下载的，过好代理关是必须的。过代理关不是你针对某个大学进行搜索，而是你积累了多少大学的代理资源。代理是有时效的，时开时关，但有一些代理可以成活很长的时间，如果你对代理资源有足够的贮备的话，进镜像超星就易于反掌。我现在收集的大学超星代理已达到 3000 多个，每天可供我用的有效代理入口至少在 50 个以上，这一切都在于平时对资源的积累。

至于一些更深层次的问题我相应新人也会有了解的一天，如果你做到了以上三点，我相信你就有用不尽的资源。

作者：lucy12345678

超星下载之路如何走？一步一脚印，忌急功近利

本帖被 acdacd 执行提前操作(2007-05-12)

超星下载之路如何走？一步一脚印，忌急功近利

初识园地：网络搜索超星破解版软件时

注册园地：为求一本英语学习辅导书时

入园初始：上天无路入地无门。求书不能，短信无应。

应对之策：自救

自救之基础：非计算机专业但能熟练使用计算机

推理：思来想去获取电子图书应该如同借阅纸质书，必须进图书馆，查询，确定书架编号，办理手续取书

具体行动：

1。寻找超星图书馆：网络资讯人人可查。利用 google, 百度搜索引擎搜索超星电子图书馆网址。

在找寻超星电子图书馆网址的过程中了解到了 ssreader.com, sslibrary.com, duxiu.com，firstdrs.com 以及各大高等院校的电子图书馆都有超星图书。其中各大高等院校的电子图书馆中的超星书量各校不等，但馆数众多因此相对比较容易找到能下载的突破口。

2。攻克高校镜像：高校入口众多，登陆条件不一。有些能毫无阻拦登陆，

有些有 ip 限制。能登陆者为首选。

在能自由登陆的高校入口中发现几种情况：检索结果能阅读下载，能阅读但不能下载，或两者都不能。如何解决下载的问题呢？旧贴中有许多方法，但我选择了添加指令法，即在连接地址中添加&candownload=1&downloadreg=0&canprint=1&。至此基本了解了超星书的下载过程及其所需参数。这是我的第一步脚印。

3。如何开发利用有 ip 限制的高校入口？旧贴中有不同的方法，我选择了试探性的 ss 号查询法和移花接木法。学会这两招完全是基于在学走第一步的过程中对快击键的观察以及旧贴中 bookish 大侠对超星书地址的解读。这是我的第二步脚印。ss 号和移花接木法的 disk 哪儿来？读秀网站查询即可。

三个月中不断来回走这两步。不断的来回，不断的阅贴，不断地思索，不断地加深领会 dchong 写的“与新手分享”贴。在这过程中也学会了翻页工具及嗅探工具以及 be 和 flashget 对高校镜像的下载方法。自我评判已达到了 zhuce2003 大侠眼中的 3 级（有 ip 限制的普通大镜像－－高级新手）或者 slonecn 大侠眼中的 4 段（会使用 BE、flashget 等第三方工具下载镜像图书），至此已经满足了我的工作学习和休闲阅读的一般所需。

4。攻克 sslibrary 入口：在找寻高校镜像入口的过程中，时不时地看到超星全库的试用通知，个别高校甚至有密码公布。于是利用 google, 百度搜索引擎搜索这些密码并试探性登陆，有些竟登陆成功。至此可阅读的范围更进了一步。如何得到这些书呢？旧贴中的首选答案有两个：嗅探截流和 be 下载。下载成功关键是有效报文（http header），获取有效报头又以嗅探截流为基础。对于超星有效报头的公开讨论园地是忌讳的，但是蛛丝马迹仍然可以在旧贴中看到。如：1）凡是能发送和接受报文的软件均可下载；2）报文是由两大部分组成；3）报文的关键指令是哪个；4）一个很简单的思路：如果例子可以阅读，那么首先把报文弄成和例子一模一样；5）得到了最初的一两页就是成功的第一步，对报文进行加加减减的美容术或者照单全收。等等。。。不断的探索如何获取报文，不断地比较分析，尝试找出报文中的共同点和不同点。按照 BE 的使用方法，在自定义报文处填入不同的报文成份进行试探性下载，终于功夫不负有心人，一天晚上终于得到了两页目录页，经过努力终于下载了第一本完整的书。现在大致了解了 BE 下载的过程，对那些蛛丝马迹有了进一步的理解。报文不可不完整，但必要时须舍得断其左膀右臂，甚至拦腰截断。这是我的第三步脚印。自我评判已达到了 slonecn 大侠眼中的 5 段（使用 BE、flashget 等第三方工具下载 sslib）。

5。下一步打算：学习代理知识

6。结论：多翻阅非加密旧贴也可找到发光的金子。一步一脚印步向光明顶。

这是我入园三个月来的学习小结。

最后感谢园地的各位网友，是你们的奉献才使我学到了这么多有用的网络知识。

既然这本书不行,换个镜象试试.
我已经试成功了.用 BE 下载图书.
报头填:
SSAuth: ******
SSRANDOM: ******
ServerID: ******
Cache-Control: no-cache

任务位置:
http://ip 网址/path/disk/